[Ignorer et passer au contenu](https://chatgpt.com/#main)

**ChatGPT**

**Se connecter**

**S'inscrire**

**Vous avez dit :**

je dois faire un projet en spring batch qui prend un fichier plat qui passe à travers un batch mapping et le converti en json step 1: chargement du fichier avec un file entete contenant id, nom, nb ligne, date et un fichier details dont lequel chaque ligne du fichier plat va etre dans une colonne (une ligne dans une case, et l'autre dans celle en dessous , etc...) puis on passe au processing ou on fait le mapping step 2: lecture configuration voici mon arborescence .idea Traitement .mvn wrapper src main java Traitement TraitementApplication resources application.properties test .gitattributes .gitignore HELP.md mvnw mvnw.cmd pom.xml

**ChatGPT a dit :**

D'accord, ton projet en Spring Batch semble très intéressant ! Si je comprends bien, voici les grandes lignes de ce que tu dois accomplir :

1. **Charger un fichier plat** : Ce fichier contient un en-tête avec des informations comme l'ID, le nom, le nombre de lignes, la date, et un fichier "détails" avec des données sous forme de lignes dans des cases.
2. **Lire et traiter la configuration** : Ici, tu vas configurer ton Batch pour qu'il sache comment lire le fichier plat et le convertir en JSON.
3. **Mapping des données** : Après avoir lu les fichiers, tu feras un "mapping" pour transformer les données dans le format souhaité (JSON dans ton cas).

**Arborescence de ton projet**

L'arborescence que tu as partagée semble standard pour un projet Spring Boot avec Spring Batch. Voici une explication rapide de chaque partie :

* .idea/ : Répertoire des configurations de ton IDE (en général pour IntelliJ IDEA).
* Traitement/ : Ton code source Java.
* .mvn/ et wrapper/ : C'est lié à Maven, un outil de gestion des dépendances et de construction du projet.
* src/main/java/Traitement/ : Contient le code Java de ton application.
* src/main/resources/ : Contient les ressources comme les fichiers de configuration (application.properties ici).
* test/ : Répertoire pour les tests unitaires.
* pom.xml : Le fichier de configuration Maven où tu définis les dépendances du projet.

**Ce que tu peux faire pour chaque étape**

1. **Chargement du fichier plat :**
   * Tu vas devoir créer un ItemReader pour lire le fichier plat. En Spring Batch, tu peux utiliser FlatFileItemReader pour lire les lignes d'un fichier texte.
   * Tu devras probablement utiliser un LineMapper pour mapper chaque ligne du fichier plat dans un objet Java.
   * Utilise DelimitedLineTokenizer pour définir comment les données sont séparées (par exemple, des virgules ou des tabulations).
2. **Traitement (Step 2 : lecture de la configuration et mapping) :**
   * Pour traiter les données, tu vas créer un ItemProcessor qui fera le mapping des données du fichier plat vers un objet Java ou directement vers un format JSON.
   * Si tu veux créer du JSON, tu peux utiliser une bibliothèque comme Jackson pour sérialiser tes objets Java en JSON.
3. **Écriture (sortie JSON) :**
   * Une fois les données traitées, tu devras créer un ItemWriter qui écrira ces données sous forme de JSON dans un fichier.
   * Spring Batch permet d'utiliser des writers comme FlatFileItemWriter ou tu peux configurer un JsonItemWriter.

**Quelques suggestions pour ta mise en œuvre**

1. **Fichier de configuration (**application.properties**)** : Tu peux y définir des propriétés comme le chemin des fichiers d'entrée et de sortie, ainsi que les autres paramètres batch comme la fréquence d'exécution, les paramètres du job, etc.

properties

CopierModifier

batch.input.file.path=path/to/input/file batch.output.file.path=path/to/output/file.json

1. **Fichiers de traitement :**
   * **Fichier d'en-tête** : Tu peux lire ce fichier séparément ou l'inclure dans ton batch, en fonction de sa structure.
   * **Fichier de détails** : Utilise un FlatFileItemReader pour ce fichier, avec un LineMapper qui transformera chaque ligne en une représentation Java.
2. **Spring Batch Job** : Voici un exemple très basique d'une structure pour ton job Spring Batch :

java

CopierModifier

@SpringBootApplication public class TraitementApplication { public static void main(String[] args) { SpringApplication.run(TraitementApplication.class, args); } @Bean public Job processFileJob(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory) { return jobBuilderFactory.get("processFileJob") .start(loadFileStep(stepBuilderFactory)) .next(processFileStep(stepBuilderFactory)) .build(); } @Bean public Step loadFileStep(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("loadFileStep") .<String, String>chunk(10) .reader(fileReader()) .processor(fileProcessor()) .writer(fileWriter()) .build(); } @Bean public ItemReader<String> fileReader() { *// Définir ton reader pour le fichier plat ici* } @Bean public ItemProcessor<String, String> fileProcessor() { *// Ton item processor ici pour effectuer le mapping* } @Bean public ItemWriter<String> fileWriter() { *// Définir un writer pour écrire le JSON ici* } }

Pour ce qui est des classes ItemReader, ItemProcessor, et ItemWriter, tu les définiras en fonction de la structure exacte de ton fichier plat et du format de sortie souhaité.

**Points de vigilance**

* **Gestion des erreurs** : Pense à bien gérer les erreurs dans ton batch, notamment en cas de ligne corrompue dans le fichier plat.
* **Performances** : Si tu traites un grand nombre de données, assure-toi que ton batch est bien configuré pour être performant (gestion des transactions, taille des chunks, etc.).
* **Tests unitaires** : N'oublie pas d'écrire des tests unitaires pour tes ItemReader, ItemProcessor, et ItemWriter pour garantir leur bon fonctionnement.

Si tu veux que je t'aide plus précisément pour une des étapes, n’hésite pas à me donner plus de détails !

**Vous avez dit :**

je veux que tu me donne un projet complet de A à Z

**ChatGPT a dit :**

Bien sûr ! Je vais te guider à travers la création d'un projet Spring Batch complet de A à Z pour transformer un fichier plat en JSON. Je vais te donner le code et les étapes nécessaires pour construire ton projet.

**Étape 1 : Préparation de l'environnement**

Assure-toi d’avoir Maven d'installé sur ta machine. Si ce n’est pas le cas, tu peux le télécharger ici : <https://maven.apache.org/download.cgi>.

Nous allons créer un projet Spring Boot avec Spring Batch, qui lira un fichier plat et générera un fichier JSON en sortie.

**Étape 2 : Structure du projet**

Ton projet aura la structure suivante :

css

CopierModifier

src

├── main

│ ├── java

│ │ └── com

│ │ └── exemple

│ │ └── traitement

│ │ ├── config

│ │ │ └── BatchConfig.java

│ │ ├── model

│ │ │ └── Detail.java

│ │ ├── processor

│ │ │ └── DetailProcessor.java

│ │ ├── reader

│ │ │ └── DetailReader.java

│ │ ├── writer

│ │ │ └── DetailWriter.java

│ │ └── TraitementApplication.java

│ ├── resources

│ │ └── application.properties

├── pom.xml

**Étape 3 : Création du fichier**pom.xml

Voici un pom.xml qui inclut toutes les dépendances nécessaires pour Spring Boot et Spring Batch, ainsi que Jackson pour la gestion du JSON :

xml

CopierModifier

<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <groupId>com.exemple</groupId> <artifactId>traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>traitement</name> <description>Un projet Spring Batch pour transformer un fichier plat en JSON</description> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>2.5.4</version> <relativePath/> *<!-- lookup parent from repository -->* </parent> <properties> <java.version>11</java.version> </properties> <dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency> <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project>

**Étape 4 : Configuration de Spring Batch**

Crée une classe de configuration pour Spring Batch dans le répertoire config.

BatchConfig.java**:**

java

CopierModifier

package com.exemple.traitement.config; import org.springframework.batch.core.BatchStatus; import org.springframework.batch.core.Job; import org.springframework.batch.core.JobParametersBuilder; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.launch.JobParametersInvalidException; import org.springframework.batch.core.launch.JobExecutionException; import org.springframework.batch.core.repository.JobRepository; import org.springframework.batch.core.repository.support.JobRepositoryFactoryBean; import org.springframework.batch.core.step.tasklet.TaskletStep; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemWriter; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.context.annotation.ComponentScan; import org.springframework.core.io.ClassPathResource; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.PassThroughLineMapper; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; @Configuration @EnableBatchProcessing @ComponentScan(basePackages = "com.exemple.traitement") public class BatchConfig { @Bean public Job processFileJob(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, ItemReader<String> fileReader, ItemProcessor<String, String> fileProcessor, ItemWriter<String> fileWriter) { return jobBuilderFactory.get("processFileJob") .start(fileStep(stepBuilderFactory, fileReader, fileProcessor, fileWriter)) .build(); } @Bean public Step fileStep(StepBuilderFactory stepBuilderFactory, ItemReader<String> fileReader, ItemProcessor<String, String> fileProcessor, ItemWriter<String> fileWriter) { return stepBuilderFactory.get("fileStep") .<String, String>chunk(10) .reader(fileReader) .processor(fileProcessor) .writer(fileWriter) .build(); } @Bean public FlatFileItemReader<String> fileReader() { FlatFileItemReader<String> reader = new FlatFileItemReader<>(); reader.setResource(new ClassPathResource("input/data.txt")); reader.setLineMapper(new PassThroughLineMapper()); return reader; } @Bean public ItemProcessor<String, String> fileProcessor() { return new ItemProcessor<String, String>() { @Override public String process(String item) { *// Traitement de la ligne : ici un exemple simple* return item; } }; } @Bean public ItemWriter<String> fileWriter() { return new ItemWriter<String>() { @Override public void write(List<? extends String> items) throws Exception { *// Convertir la liste en JSON* ObjectMapper mapper = new ObjectMapper(); mapper.writeValue(new File("output/result.json"), items); } }; } }

**Étape 5 : Application principale**

TraitementApplication.java**:**

java

CopierModifier

package com.exemple.traitement; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class TraitementApplication { public static void main(String[] args) { SpringApplication.run(TraitementApplication.class, args); } }

**Étape 6 : Application Properties**

Dans src/main/resources/application.properties, configure les chemins des fichiers d'entrée et de sortie :

properties

CopierModifier

batch.input.file.path=src/main/resources/input/data.txt batch.output.file.path=src/main/resources/output/result.json

**Étape 7 : Fichier d’entrée (input/data.txt)**

Voici un exemple simple de fichier d’entrée data.txt :

CopierModifier

1,Jean Dupont

2,Marie Martin

3,Paul Durand

**Étape 8 : Création des classes de modèle (Optionnel)**

Tu peux créer une classe modèle si tu veux mapper directement les données dans des objets Java.

Detail.java**:**

java

CopierModifier

package com.exemple.traitement.model; public class Detail { private String id; private String name; *// Getters et setters* }

**Étape 9 : Lancer l'application**

Ensuite, tu peux exécuter ton application à partir de la ligne de commande :

bash

CopierModifier

mvn spring-boot:run

Une fois le batch exécuté, tu devrais trouver ton fichier JSON généré dans output/result.json.

Voilà, tu as maintenant un projet Spring Batch complet pour lire un fichier plat et le transformer en JSON. Si tu veux plus de détails sur une partie particulière, ou si tu rencontres des problèmes, n'hésite pas à me le dire !

**Vous avez dit :**

je veux un code complet sachant que le fichier plat est écrit comme suivant name famillyname age ville

**ChatGPT a dit :**

D'accord, je vois que tu veux que ton fichier plat contienne des colonnes spécifiques avec les données suivantes : name, familyname, age, et ville. On va donc construire un projet Spring Batch complet qui lira ce fichier et le transformera en JSON.

Voici un exemple complet, détaillé, avec toutes les étapes que tu peux suivre.

**Structure du projet**

css

CopierModifier

src

├── main

│ ├── java

│ │ └── com

│ │ └── exemple

│ │ └── traitement

│ │ ├── config

│ │ │ └── BatchConfig.java

│ │ ├── model

│ │ │ └── Person.java

│ │ ├── processor

│ │ │ └── PersonProcessor.java

│ │ ├── reader

│ │ │ └── PersonReader.java

│ │ ├── writer

│ │ │ └── PersonWriter.java

│ │ └── TraitementApplication.java

│ ├── resources

│ │ └── input

│ │ └── people.txt

│ │ └── output

│ │ └── result.json

│ ├── application.properties

├── pom.xml

**1. Fichier**pom.xml

Voici le fichier pom.xml pour configurer ton projet Maven avec Spring Boot, Spring Batch, et Jackson pour la gestion du JSON.

xml

CopierModifier

<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <groupId>com.exemple</groupId> <artifactId>traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>traitement</name> <description>Un projet Spring Batch pour transformer un fichier plat en JSON</description> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>2.5.4</version> <relativePath/> *<!-- lookup parent from repository -->* </parent> <properties> <java.version>11</java.version> </properties> <dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency> <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project>

**2. Fichier de configuration**BatchConfig.java

Ce fichier contient la configuration de Spring Batch pour ton job.

java

CopierModifier

package com.exemple.traitement.config; import com.exemple.traitement.model.Person; import com.exemple.traitement.processor.PersonProcessor; import com.exemple.traitement.reader.PersonReader; import com.exemple.traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemWriter; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job processFileJob(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, ItemReader<Person> fileReader, ItemProcessor<Person, Person> fileProcessor, ItemWriter<Person> fileWriter) { return jobBuilderFactory.get("processFileJob") .start(fileStep(stepBuilderFactory, fileReader, fileProcessor, fileWriter)) .build(); } @Bean public Step fileStep(StepBuilderFactory stepBuilderFactory, ItemReader<Person> fileReader, ItemProcessor<Person, Person> fileProcessor, ItemWriter<Person> fileWriter) { return stepBuilderFactory.get("fileStep") .<Person, Person>chunk(10) .reader(fileReader) .processor(fileProcessor) .writer(fileWriter) .build(); } @Bean public ItemReader<Person> fileReader() { return new PersonReader(); } @Bean public ItemProcessor<Person, Person> fileProcessor() { return new PersonProcessor(); } @Bean public ItemWriter<Person> fileWriter() { return new PersonWriter(); } }

**3. Modèle**Person.java

Le modèle Person représente une personne avec des champs name, familyname, age et ville.

java

CopierModifier

package com.exemple.traitement.model; public class Person { private String name; private String familyname; private int age; private String ville; *// Getters et setters* public String getName() { return name; } public void setName(String name) { this.name = name; } public String getFamilyname() { return familyname; } public void setFamilyname(String familyname) { this.familyname = familyname; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } }

**4. Lecteur**PersonReader.java

Le lecteur va lire les lignes du fichier plat, avec un tokenizer pour décomposer les colonnes séparées par des espaces ou des tabulations.

java

CopierModifier

package com.exemple.traitement.reader; import com.exemple.traitement.model.Person; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; public class PersonReader extends FlatFileItemReader<Person> { public PersonReader() { this.setResource(new ClassPathResource("input/people.txt")); *// Tokenizer qui sépare les champs par des espaces* DelimitedLineTokenizer tokenizer = new DelimitedLineTokenizer(); tokenizer.setDelimiter(" "); tokenizer.setNames("name", "familyname", "age", "ville"); *// Mapper les champs vers l'objet Person* BeanWrapperFieldSetMapper<Person> fieldSetMapper = new BeanWrapperFieldSetMapper<>(); fieldSetMapper.setTargetType(Person.class); setLineMapper((line, lineNumber) -> { return fieldSetMapper.mapFieldSet(tokenizer.tokenize(line)); }); } }

**5. Processor**PersonProcessor.java

Le PersonProcessor peut être utilisé pour ajouter des logiques de transformation, mais dans cet exemple, il s'agit d'un simple "pass-through" qui ne modifie pas les données.

java

CopierModifier

package com.exemple.traitement.processor; import com.exemple.traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) { *// Ajoute toute logique de transformation ici si nécessaire* return item; } }

**6. Writer**PersonWriter.java

Le PersonWriter va écrire les données dans un fichier JSON à l'aide de la bibliothèque Jackson.

java

CopierModifier

package com.exemple.traitement.writer; import com.exemple.traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { @Override public void write(List<? extends Person> items) throws IOException { ObjectMapper objectMapper = new ObjectMapper(); objectMapper.writeValue(new File("output/result.json"), items); } }

**7. Fichier d'entrée**people.txt

Le fichier d'entrée, people.txt, contient des données comme suit :

nginx

CopierModifier

John Doe 30 Paris

Jane Smith 25 Lyon

Paul Martin

**Vous avez dit :**

je veux que tu continu sur ce code package com.exemple.traitement.config; import org.springframework.batch.core.BatchStatus; import org.springframework.batch.core.Job; import org.springframework.batch.core.JobParametersBuilder; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.launch.JobParametersInvalidException; import org.springframework.batch.core.launch.JobExecutionException; import org.springframework.batch.core.repository.JobRepository; import org.springframework.batch.core.repository.support.JobRepositoryFactoryBean; import org.springframework.batch.core.step.tasklet.TaskletStep; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemWriter; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.context.annotation.ComponentScan; import org.springframework.core.io.ClassPathResource; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.PassThroughLineMapper; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; @Configuration @EnableBatchProcessing @ComponentScan(basePackages = "com.exemple.traitement") public class BatchConfig { @Bean public Job processFileJob(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, ItemReader<String> fileReader, ItemProcessor<String, String> fileProcessor, ItemWriter<String> fileWriter) { return jobBuilderFactory.get("processFileJob") .start(fileStep(stepBuilderFactory, fileReader, fileProcessor, fileWriter)) .build(); } @Bean public Step fileStep(StepBuilderFactory stepBuilderFactory, ItemReader<String> fileReader, ItemProcessor<String, String> fileProcessor, ItemWriter<String> fileWriter) { return stepBuilderFactory.get("fileStep") .<String, String>chunk(10) .reader(fileReader) .processor(fileProcessor) .writer(fileWriter) .build(); } @Bean public FlatFileItemReader<String> fileReader() { FlatFileItemReader<String> reader = new FlatFileItemReader<>(); reader.setResource(new ClassPathResource("input/data.txt")); reader.setLineMapper(new PassThroughLineMapper()); return reader; } @Bean public ItemProcessor<String, String> fileProcessor() { return new ItemProcessor<String, String>() { @Override public String process(String item) { // Traitement de la ligne : ici un exemple simple return item; } }; } @Bean public ItemWriter<String> fileWriter() { return new ItemWriter<String>() { @Override public void write(List<? extends String> items) throws Exception { // Convertir la liste en JSON ObjectMapper mapper = new ObjectMapper(); mapper.writeValue(new File("output/result.json"), items); } }; } }

**ChatGPT a dit :**

D'accord, tu veux continuer à partir de ce code pour faire en sorte qu'il fonctionne avec un fichier plat structuré contenant les colonnes name, familyname, age, et ville, et pour que ces données soient transformées en JSON. Nous allons adapter ton code pour que la lecture du fichier se fasse en utilisant un ItemReader approprié, puis les données seront traitées et écrites en format JSON avec un ItemProcessor et un ItemWriter.

Je vais également intégrer la gestion de l'objet Person et ajouter la gestion du fichier d'entrée.

**Étape 1 : Mise à jour de la classe**BatchConfig.java

La classe BatchConfig doit être mise à jour pour inclure un ItemReader, un ItemProcessor, et un ItemWriter qui manipulent des objets de type Person (au lieu de String).

**Code de la classe**BatchConfig.java**mise à jour :**

java

CopierModifier

package com.exemple.traitement.config; import com.exemple.traitement.model.Person; import com.exemple.traitement.processor.PersonProcessor; import com.exemple.traitement.reader.PersonReader; import com.exemple.traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemWriter; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job processFileJob(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, ItemReader<Person> fileReader, ItemProcessor<Person, Person> fileProcessor, ItemWriter<Person> fileWriter) { return jobBuilderFactory.get("processFileJob") .start(fileStep(stepBuilderFactory, fileReader, fileProcessor, fileWriter)) .build(); } @Bean public Step fileStep(StepBuilderFactory stepBuilderFactory, ItemReader<Person> fileReader, ItemProcessor<Person, Person> fileProcessor, ItemWriter<Person> fileWriter) { return stepBuilderFactory.get("fileStep") .<Person, Person>chunk(10) .reader(fileReader) .processor(fileProcessor) .writer(fileWriter) .build(); } @Bean public ItemReader<Person> fileReader() { return new PersonReader(); *// Utilisation de notre lecteur personnalisé* } @Bean public ItemProcessor<Person, Person> fileProcessor() { return new PersonProcessor(); *// Utilisation de notre processeur personnalisé* } @Bean public ItemWriter<Person> fileWriter() { return new PersonWriter(); *// Utilisation de notre écrivain personnalisé* } }

**Étape 2 : Création du modèle**Person.java

Ce modèle contiendra les propriétés name, familyname, age, et ville, ainsi que les getters et setters pour chaque propriété.

java

CopierModifier

package com.exemple.traitement.model; public class Person { private String name; private String familyname; private int age; private String ville; *// Getters et setters* public String getName() { return name; } public void setName(String name) { this.name = name; } public String getFamilyname() { return familyname; } public void setFamilyname(String familyname) { this.familyname = familyname; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } }

**Étape 3 : Création du**PersonReader.java

Le PersonReader lira les lignes du fichier plat et les transformera en objets Person. Le fichier est supposé être structuré avec des colonnes séparées par des espaces ou des tabulations.

java

CopierModifier

package com.exemple.traitement.reader; import com.exemple.traitement.model.Person; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; import org.springframework.core.io.ClassPathResource; public class PersonReader extends FlatFileItemReader<Person> { public PersonReader() { this.setResource(new ClassPathResource("input/people.txt")); *// Chemin du fichier d'entrée* *// Tokenizer qui sépare les champs par des espaces (ou des tabulations)* DelimitedLineTokenizer tokenizer = new DelimitedLineTokenizer(); tokenizer.setDelimiter("\\s+"); *// Utilisation d'un espace ou tabulation comme délimiteur* tokenizer.setNames("name", "familyname", "age", "ville"); *// Mapper les champs vers l'objet Person* BeanWrapperFieldSetMapper<Person> fieldSetMapper = new BeanWrapperFieldSetMapper<>(); fieldSetMapper.setTargetType(Person.class); setLineMapper((line, lineNumber) -> { return fieldSetMapper.mapFieldSet(tokenizer.tokenize(line)); }); } }

**Étape 4 : Création du**PersonProcessor.java

Le PersonProcessor peut être utilisé pour ajouter des logiques de transformation ou de validation avant d'écrire les données. Pour cet exemple, nous allons juste laisser passer les objets tels quels.

java

CopierModifier

package com.exemple.traitement.processor; import com.exemple.traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) { *// On peut ajouter une logique de transformation ici si nécessaire* return item; *// Ne change rien dans cet exemple* } }

**Étape 5 : Création du**PersonWriter.java

Le PersonWriter va prendre les objets Person et les écrire dans un fichier JSON à l'aide de la bibliothèque Jackson.

java

CopierModifier

package com.exemple.traitement.writer; import com.exemple.traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { @Override public void write(List<? extends Person> items) throws IOException { *// Crée une instance d'ObjectMapper pour convertir en JSON* ObjectMapper objectMapper = new ObjectMapper(); objectMapper.writeValue(new File("output/result.json"), items); } }

**Étape 6 : Fichier d'entrée**people.txt

Voici un exemple de fichier d'entrée people.txt contenant les informations des personnes. Ce fichier doit être dans le dossier src/main/resources/input/.

nginx

CopierModifier

John Doe 30 Paris

Jane Smith 25 Lyon

Paul Martin 40 Marseille

**Étape 7 : Exécution de l'application**

1. **Configuration de**application.properties**(facultatif)** :

Tu peux ajouter un fichier src/main/resources/application.properties si tu veux configurer des paramètres, comme les chemins des fichiers d'entrée et de sortie.

properties

CopierModifier

batch.input.file.path=src/main/resources/input/people.txt batch.output.file.path=src/main/resources/output/result.json

1. **Lancer l'application** :

Tu peux maintenant lancer ton application Spring Boot avec la commande suivante :

bash

CopierModifier

mvn spring-boot:run

1. **Vérification des résultats** :

Une fois l'exécution terminée, tu devrais trouver le fichier result.json dans le dossier src/main/resources/output/ contenant les données en JSON.

Exemple de result.json généré :

json

CopierModifier

[ { "name": "John", "familyname": "Doe", "age": 30, "ville": "Paris" }, { "name": "Jane", "familyname": "Smith", "age": 25, "ville": "Lyon" }, { "name": "Paul", "familyname": "Martin", "age": 40, "ville": "Marseille" } ]

**Conclusion**

Le projet Spring Batch est maintenant prêt. Il lit un fichier plat, transforme les lignes en objets Person, et les écrit en JSON. Tu peux ajuster la logique selon tes besoins, ajouter de la validation, ou personnaliser le traitement des données.

**Vous avez dit :**

le fichier plat s'écrit comme 01 header(avec nb de ligne , nom fichier...) 02 0001 contenu 02 0002 contenu . . . . . . . 02 5000 contenu 03 footer (nb de ligne ...)

**ChatGPT a dit :**

D'accord, maintenant que tu m'as précisé que le fichier plat contient une structure avec un **header**, des **données** (ou contenu), et un **footer**, nous devons adapter le traitement dans le projet Spring Batch pour tenir compte de cette structure.

**Structure du fichier plat**

Ton fichier d'entrée semble avoir la structure suivante :

* **Ligne 1** : Header, contenant des informations sur le fichier (nombre de lignes, nom du fichier, etc.)
* **Lignes 2 à N** : Données qui représentent les enregistrements individuels (contenu)
* **Dernière ligne** : Footer, contenant des informations comme le nombre de lignes de données

Voici un exemple de fichier plat (data.txt) :

yaml

CopierModifier

01 header | nb lignes: 5000 | nom fichier: data.txt

02 0001 John Doe 30 Paris

02 0002 Jane Smith 25 Lyon

...

02 5000 Paul Martin 40 Marseille

03 footer | nb lignes: 5000

**Modifications nécessaires :**

1. **Lire les lignes du fichier (Header, Data, Footer)** :
   * On doit ignorer les lignes de type 01 et 03 (Header et Footer) pendant le traitement des données.
   * Nous devons capturer les lignes de type 02 comme données.
2. **Adapter le**PersonReader**pour gérer cette structure**.
3. **Faire en sorte que le**PersonProcessor**et**PersonWriter**traitent les données correctement**.

**Étape 1 : Mise à jour du modèle**Person

Le modèle Person reste le même que précédemment, puisqu'on travaille toujours avec des données comme name, familyname, age, et ville.

java

CopierModifier

package com.exemple.traitement.model; public class Person { private String name; private String familyname; private int age; private String ville; *// Getters et setters* public String getName() { return name; } public void setName(String name) { this.name = name; } public String getFamilyname() { return familyname; } public void setFamilyname(String familyname) { this.familyname = familyname; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } }

**Étape 2 : Mise à jour du**PersonReader

Nous allons maintenant adapter le PersonReader pour ignorer les lignes 01 (Header) et 03 (Footer), et ne lire que les lignes 02 (les données réelles).

Voici un exemple de PersonReader mis à jour :

java

CopierModifier

package com.exemple.traitement.reader; import com.exemple.traitement.model.Person; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; import org.springframework.batch.item.file.transform.LineTokenizer; import org.springframework.core.io.ClassPathResource; public class PersonReader extends FlatFileItemReader<Person> { public PersonReader() { this.setResource(new ClassPathResource("input/data.txt")); *// Fichier d'entrée* *// Tokenizer pour analyser chaque ligne* DelimitedLineTokenizer tokenizer = new DelimitedLineTokenizer(); tokenizer.setDelimiter("\\s+"); *// Espace comme délimiteur* tokenizer.setNames("recordType", "id", "name", "familyname", "age", "ville"); *// Mapper les champs vers l'objet Person* BeanWrapperFieldSetMapper<Person> fieldSetMapper = new BeanWrapperFieldSetMapper<>(); fieldSetMapper.setTargetType(Person.class); setLineMapper((line, lineNumber) -> { *// Filtrage des lignes : ignorer les lignes 01 (header) et 03 (footer)* if (line.startsWith("01") || line.startsWith("03")) { return null; *// Ignore la ligne* } *// Si la ligne commence par "02", c'est une ligne de données* return fieldSetMapper.mapFieldSet(tokenizer.tokenize(line)); }); } }

**Explication du**PersonReader**:**

1. **Lecture du fichier** : Le fichier data.txt est lu à partir du chemin src/main/resources/input/.
2. **Tokenizer** : Un DelimitedLineTokenizer est utilisé pour découper chaque ligne en plusieurs champs, en utilisant les espaces comme délimiteurs. On s'attend à avoir des colonnes : recordType, id, name, familyname, age, ville.
3. **Filtrage des lignes** : On ignore les lignes qui commencent par 01 et 03 (header et footer). Les lignes commençant par 02 sont celles qui contiennent les données valides, donc elles sont transformées en objets Person.

**Étape 3 : Mise à jour du**PersonProcessor

Le PersonProcessor n'a pas besoin de modifications significatives. Il peut rester tel quel, puisque les données que nous recevons sont déjà des objets Person.

java

CopierModifier

package com.exemple.traitement.processor; import com.exemple.traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) { *// On peut ajouter une logique de transformation ici si nécessaire* return item; *// Pas de transformation pour l'instant* } }

**Étape 4 : Mise à jour du**PersonWriter

Le PersonWriter reste également inchangé par rapport à ce qu'on a déjà vu, il écrit toujours les objets Person dans un fichier JSON.

java

CopierModifier

package com.exemple.traitement.writer; import com.exemple.traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { @Override public void write(List<? extends Person> items) throws IOException { *// Crée une instance d'ObjectMapper pour convertir en JSON* ObjectMapper objectMapper = new ObjectMapper(); objectMapper.writeValue(new File("output/result.json"), items); } }

**Étape 5 : Mise à jour de la configuration**BatchConfig

Enfin, la classe de configuration BatchConfig doit être mise à jour pour utiliser ces nouveaux composants (Reader, Processor, Writer).

java

CopierModifier

package com.exemple.traitement.config; import com.exemple.traitement.model.Person; import com.exemple.traitement.processor.PersonProcessor; import com.exemple.traitement.reader.PersonReader; import com.exemple.traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemWriter; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job processFileJob(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, ItemReader<Person> fileReader, ItemProcessor<Person, Person> fileProcessor, ItemWriter<Person> fileWriter) { return jobBuilderFactory.get("processFileJob") .start(fileStep(stepBuilderFactory, fileReader, fileProcessor, fileWriter)) .build(); } @Bean public Step fileStep(StepBuilderFactory stepBuilderFactory, ItemReader<Person> fileReader, ItemProcessor<Person, Person> fileProcessor, ItemWriter<Person> fileWriter) { return stepBuilderFactory.get("fileStep") .<Person, Person>chunk(10) .reader(fileReader) .processor(fileProcessor) .writer(fileWriter) .build(); } @Bean public ItemReader<Person> fileReader() { return new PersonReader(); *// Utilisation du lecteur qui gère le header/footer* } @Bean public ItemProcessor<Person, Person> fileProcessor() { return new PersonProcessor(); *// Pas de transformation, simplement un passage* } @Bean public ItemWriter<Person> fileWriter() { return new PersonWriter(); *// Écriture en JSON* } }

**Étape 6 : Exemple de fichier d'entrée**data.txt

Voici un exemple du fichier d'entrée data.txt qui correspond à la structure que tu m'as donnée :

yaml

CopierModifier

01 header | nb lignes: 5000 | nom fichier: data.txt

02 0001 John Doe 30 Paris

02 0002 Jane Smith 25 Lyon

...

02 5000 Paul Martin 40 Marseille

03 footer | nb lignes: 5000

**Étape 7 : Exécution du projet**

1. **Lancer l'application** :

Tu peux maintenant lancer ton application avec la commande suivante :

bash

CopierModifier

mvn spring-boot:run

1. **Vérification des résultats** :

Le fichier result.json sera généré dans le dossier output/, avec un format JSON contenant les objets Person.

Exemple de result.json

**Vous avez dit :**

01 header | nb lignes: 5000 | nom fichier: data.txt 01 (c'est le header qui contient nb de ligne et le nom du fichier ) tout sans séparation on travail avec la position par exemple de0 à 5 c'est la ou on note 01 de 6 à 15 (clé de ligne ex:00001) de 16 à 48 contenu quelconque

**ChatGPT a dit :**

Merci pour la précision ! Dans ce cas, le fichier plat a une structure fixe, et il semble que chaque ligne ait des positions spécifiques pour chaque champ. Voici comment nous allons procéder pour gérer cela :

* **Header (**01**)** : Occupe la première partie de la ligne, de 0 à 5 pour 01, et les informations de ligne et nom de fichier se trouvent dans d'autres parties de la ligne.
* **Data (**02**)** : Chaque ligne commence par un identifiant (00001, 00002, ...) et un contenu (name, familyname, age, ville), avec des champs positionnés dans des colonnes fixes.
* **Footer (**03**)** : Donne des informations supplémentaires, mais sera ignoré pendant le traitement des données.

**Structure du fichier plat :**

* **Ligne Header** :
  + 01 (de 0 à 5)
  + Clé de ligne (de 6 à 15)
  + Contenu (de 16 à 48) — ici, on extrait des informations comme le nombre de lignes et le nom du fichier.
* **Ligne de données** :
  + 02 (de 0 à 5)
  + Clé de ligne (de 6 à 15)
  + Contenu (de 16 à 48) — avec les valeurs name, familyname, age, ville.
* **Footer** :
  + 03 (de 0 à 5)
  + Contenu similaire aux informations de la première ligne du header.

**Étape 1 : Adapter le**PersonReader**pour gérer des positions fixes**

Nous allons lire les fichiers en prenant en compte les **positions fixes** des champs dans chaque ligne. Nous allons créer un lecteur qui découpe la ligne en fonction des indices que tu as fournis, puis il mappe les données dans un objet Person.

**Exemple de**PersonReader.java**:**

java

CopierModifier

package com.exemple.traitement.reader; import com.exemple.traitement.model.Person; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper; import org.springframework.batch.item.file.transform.LineTokenizer; import org.springframework.batch.item.file.transform.FixedWidthTokenizer; import org.springframework.core.io.ClassPathResource; public class PersonReader extends FlatFileItemReader<Person> { public PersonReader() { *// Le fichier d'entrée* this.setResource(new ClassPathResource("input/data.txt")); *// Créer un tokenizer avec des largeurs fixes pour chaque champ* FixedWidthTokenizer tokenizer = new FixedWidthTokenizer(); tokenizer.setNames("recordType", "lineKey", "name", "familyname", "age", "ville"); *// Définir les positions des colonnes* tokenizer.setColumns( new Range(0, 5), *// recordType (01, 02, ou 03)* new Range(6, 15), *// lineKey (ex : 00001, 00002, ...)* new Range(16, 48), *// name* new Range(49, 79), *// familyname* new Range(80, 82), *// age* new Range(83, 113) *// ville* ); *// Mapper les données dans un objet Person* BeanWrapperFieldSetMapper<Person> fieldSetMapper = new BeanWrapperFieldSetMapper<>(); fieldSetMapper.setTargetType(Person.class); *// Configurer le LineMapper* setLineMapper((line, lineNumber) -> { *// Si la ligne est un header (01) ou un footer (03), on l'ignore* if (line.startsWith("01") || line.startsWith("03")) { return null; *// On ignore le header et le footer* } *// Pour la ligne de données (02), on crée un objet Person* return fieldSetMapper.mapFieldSet(tokenizer.tokenize(line)); }); } }

**Explication :**

1. **FixedWidthTokenizer** : Ce tokenizer est adapté pour traiter des fichiers avec des champs positionnés à des indices fixes. On spécifie les positions des colonnes avec Range(0, 5), Range(6, 15), etc.
2. **Mapping des données** : Une fois la ligne lue, elle est découpée en un ensemble de champs que l'on mappe sur un objet Person.

**Étape 2 : Adapter la classe**Person**pour recevoir les données**

La classe Person reste simple. Elle doit correspondre aux champs que nous lisons à partir du fichier plat :

java

CopierModifier

package com.exemple.traitement.model; public class Person { private String name; private String familyname; private int age; private String ville; *// Getters et setters* public String getName() { return name; } public void setName(String name) { this.name = name; } public String getFamilyname() { return familyname; } public void setFamilyname(String familyname) { this.familyname = familyname; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } }

**Étape 3 : Configuration de Spring Batch**

Nous devons ajuster la configuration de Spring Batch pour que les ItemReader, ItemProcessor et ItemWriter fonctionnent avec les données correctement lues à partir du fichier.

**BatchConfig.java (mise à jour) :**

java

CopierModifier

package com.exemple.traitement.config; import com.exemple.traitement.model.Person; import com.exemple.traitement.processor.PersonProcessor; import com.exemple.traitement.reader.PersonReader; import com.exemple.traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemWriter; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job processFileJob(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, ItemReader<Person> fileReader, ItemProcessor<Person, Person> fileProcessor, ItemWriter<Person> fileWriter) { return jobBuilderFactory.get("processFileJob") .start(fileStep(stepBuilderFactory, fileReader, fileProcessor, fileWriter)) .build(); } @Bean public Step fileStep(StepBuilderFactory stepBuilderFactory, ItemReader<Person> fileReader, ItemProcessor<Person, Person> fileProcessor, ItemWriter<Person> fileWriter) { return stepBuilderFactory.get("fileStep") .<Person, Person>chunk(10) .reader(fileReader) .processor(fileProcessor) .writer(fileWriter) .build(); } @Bean public ItemReader<Person> fileReader() { return new PersonReader(); *// Utilisation du lecteur qui gère le header/footer et le découpage par position* } @Bean public ItemProcessor<Person, Person> fileProcessor() { return new PersonProcessor(); *// Pas de transformation, simplement un passage* } @Bean public ItemWriter<Person> fileWriter() { return new PersonWriter(); *// Écriture en JSON* } }

**Étape 4 : Le fichier d'entrée (**data.txt**)**

Voici un exemple de fichier d'entrée data.txt, avec les positions fixes comme tu l'as décrit :

yaml

CopierModifier

01header | nb lignes: 5000 | nom fichier: data.txt

02 00001 John Doe 30 Paris

02 00002 Jane Smith 25 Lyon

02 00003 Paul Martin 40 Marseille

...

03footer | nb lignes: 5000

Les différentes parties sont positionnées comme suit :

* 01 dans les 5 premiers caractères.
* 00001, 00002, etc., dans les caractères 6 à 15.
* Le nom, prénom, âge et ville dans les colonnes respectives (positionnées selon ton format).

**Étape 5 : Exécution**

Tu peux maintenant exécuter ton application comme d'habitude :

bash

CopierModifier

mvn spring-boot:run

À la fin du traitement, le fichier result.json sera généré avec les données en format JSON, et il contiendra des objets Person.

Exemple de result.json :

json

CopierModifier

[ { "name": "John", "familyname": "Doe", "age": 30, "ville": "Paris" }, { "name": "Jane", "familyname": "Smith", "age": 25, "ville": "Lyon" }, { "name": "Paul", "familyname": "Martin", "age": 40, "ville": "Marseille" } ]

**Conclusion**

Le fichier plat avec des positions fixes est maintenant correctement lu, transformé en objets Person, et les données sont exportées en format JSON. Le FixedWidthTokenizer nous permet de gérer des fichiers

**Vous avez dit :**

<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>3.4.4</version> <relativePath/> <!-- lookup parent from repository --> </parent> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>Traitement</name> <description>Demo project for Spring Boot</description> <url/> <licenses> <license/> </licenses> <developers> <developer/> </developers> <scm> <connection/> <developerConnection/> <tag/> <url/> </scm> <properties> <java.version>17</java.version> </properties> <dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency> <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <optional>true</optional> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> <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-data-jpa</artifactId> <version>3.4.4</version> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> <version>3.4.4</version> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <configuration> <annotationProcessorPaths> <path> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </path> </annotationProcessorPaths> </configuration> </plugin> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> <configuration> <excludes> <exclude> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </exclude> </excludes> </configuration> </plugin> </plugins> </build> </project> voici mon pom je veux que tu me donne un pom avec les dependencies pour ce projet

**ChatGPT a dit :**

Voici un fichier pom.xml mis à jour pour ton projet Spring Boot avec Spring Batch et les dépendances appropriées pour manipuler un fichier plat, et les outils de traitement par lot, incluant également les dépendances pour PostgreSQL et Lombok (si nécessaire).

pom.xml**pour ton projet :**

xml

CopierModifier

<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>3.4.4</version> <relativePath/> *<!-- lookup parent from repository -->* </parent> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>Traitement</name> <description>Demo project for Spring Boot with Batch processing</description> <properties> <java.version>17</java.version> </properties> <dependencies> *<!-- Spring Boot Starter for Batch Processing -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> *<!-- Spring Boot Starter for JPA and Database Access -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> *<!-- PostgreSQL Database Driver -->* <dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency> *<!-- Lombok for reducing boilerplate code (optional, you can remove if not needed) -->* <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <optional>true</optional> </dependency> *<!-- Spring Boot Starter Test for Unit and Integration Tests -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> *<!-- Spring Batch Test (for testing batch jobs) -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-test</artifactId> <scope>test</scope> </dependency> *<!-- Jackson for JSON processing -->* <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> *<!-- Fixed Width Tokenizer for handling fixed-width files (you can process files with specific column positions) -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> </dependency> *<!-- Spring Boot Maven Plugin -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </dependency> </dependencies> <build> <plugins> *<!-- Maven Compiler Plugin (to use Java 17) -->* <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <configuration> <source>17</source> <target>17</target> <annotationProcessorPaths> <path> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </path> </annotationProcessorPaths> </configuration> </plugin> *<!-- Spring Boot Maven Plugin to build and run Spring Boot applications -->* <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project>

**Explications des principales dépendances :**

1. **Spring Batch** (spring-boot-starter-batch et spring-batch-core): Ce starter contient les bibliothèques nécessaires pour gérer les traitements par lot (batch jobs) comme lire, traiter et écrire les données dans un fichier plat ou une base de données.
2. **Spring Data JPA** (spring-boot-starter-data-jpa): Cette dépendance permet de connecter Spring Boot à une base de données relationnelle comme PostgreSQL via JPA (Java Persistence API).
3. **PostgreSQL** (postgresql): Le pilote de base de données pour PostgreSQL, en scope runtime, il sera utilisé pour la connexion à la base de données lors de l'exécution.
4. **Lombok** (lombok): Bien qu'il soit optionnel, Lombok réduit considérablement le code boilerplate pour les getters, setters, toString, etc.
5. **Spring Batch Test** (spring-batch-test): Utilisé pour les tests unitaires et d'intégration des jobs batch, ce module est très utile pour tester vos jobs avant leur déploiement.
6. **Jackson** (jackson-databind): Une bibliothèque utilisée pour convertir les objets Java en JSON et vice versa, ce qui est utile si tu souhaites convertir tes résultats en JSON.
7. **Maven Compiler Plugin** : Configure la version de Java utilisée pour compiler le projet, ici Java 17.
8. **Spring Boot Maven Plugin** : Permet de créer des artefacts exécutable JAR ou WAR et d'exécuter l'application Spring Boot à partir de Maven avec la commande mvn spring-boot:run.

**Ajout de la configuration pour PostgreSQL :**

Si tu veux configurer la connexion à PostgreSQL dans application.properties (ou application.yml), voici un exemple de ce que tu pourrais ajouter :

properties

CopierModifier

spring.datasource.url=jdbc:postgresql://localhost:5432/tondb spring.datasource.username=tonuser spring.datasource.password=tonmotdepasse spring.datasource.driver-class-name=org.postgresql.Driver spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect spring.jpa.hibernate.ddl-auto=update

**Conclusion :**

Ce fichier pom.xml contient toutes les dépendances nécessaires pour ton projet Spring Batch qui va traiter les fichiers plats et stocker les résultats dans une base de données ou un autre format, comme JSON. Il inclut également la gestion des tests et la configuration de la base de données PostgreSQL.

**Vous avez dit :**

Neha Amina 51 1029 Casablanca Qasimi Youssef 58 548 Tanger Hamid Hafsa 25 370 Fès Neha Khalid 52 1095 Casablanca Benjelloun Fatima 58 452 Alexandrie Hiba Fatima 43 366 Marrakech Neha Omar 55 764 Le Caire Mansour Youssef 40 1097 Alexandrie Hamid Leila 49 721 Tunis Mansour Leila 23 952 Marrakech Bouazza Ahmed 35 223 Dubaï Zahir Hafsa 48 647 Marrakech Oualid Ahmed 26 710 Marrakech Cherkaoui Omar 36 499 Rabat Cherkaoui Youssef 43 573 Casablanca Hiba Amina 46 448 Marrakech Bouazza Leila 52 794 Fès Alami Omar 55 231 Le Caire Zahir Fatima 49 784 Rabat Cherkaoui Mohamed 42 707 Dubaï Benjelloun Zahra 30 726 Casablanca Cherkaoui Amina 32 382 Le Caire Idrissi Omar 35 429 Tunis Neha Youssef 20 744 Fès Neha Ahmed 59 697 Le Caire Bouazza Khalid 47 898 Casablanca Oualid Mohamed 56 619 Dubaï Bouazza Fatima 20 666 Dubaï Hiba Leila 24 789 Casablanca Hamid Omar 56 435 Tanger Hamid Mohamed 58 703 Casablanca Alami Khalid 34 193 Casablanca Zahir Hafsa 30 257 Dubaï Hamid Khalid 27 525 Casablanca Zahir Youssef 43 777 Tunis Idrissi Omar 44 698 Rabat Saadi Hafsa 26 501 Le Caire Zahir Khalid 24 795 Tunis Hamid Khalid 52 1023 Casablanca Oualid Omar 35 836 Le Caire Alami Youssef 53 157 Alexandrie Qasimi Mohamed 35 101 Dubaï Hiba Youssef 24 970 Casablanca Hiba Hafsa 22 131 Riyad Zahir Mohamed 35 654 Dubaï Saadi Fatima 39 555 Riyad Cherkaoui Hafsa 20 320 Riyad Neha Amina 29 145 Rabat Zahir Omar 35 120 Riyad Bouazza Ahmed 59 691 Le Caire Alami Mohamed 35 1030 Rabat Alami Fatima 33 321 Riyad Bouazza Leila 42 537 Fès Hamid Amina 34 944 Tanger Neha Khalid 58 824 Tunis Mansour Fatima 34 564 Marrakech Neha Khalid 29 496 Rabat Hamid Zahra 22 757 Riyad Oualid Mohamed 21 367 Alexandrie Bouazza Fatima 50 248 Marrakech Cherkaoui Youssef 53 550 Alexandrie Neha Youssef 55 431 Rabat Benjelloun Mohamed 25 521 Le Caire Hiba Khalid 41 224 Le Caire Mansour Fatima 20 621 Rabat Mansour Fatima 38 181 Marrakech Cherkaoui Ahmed 49 930 Casablanca Hiba Leila 34 252 Rabat Hiba Mohamed 41 891 Dubaï Idrissi Mohamed 42 828 Casablanca Bouazza Amina 34 998 Le Caire Mansour Mohamed 44 600 Alexandrie Idrissi Ahmed 24 414 Alexandrie Hamid Leila 38 958 Le Caire Hamid Ahmed 38 176 Dubaï Idrissi Mohamed 49 420 Fès Mansour Amina 53 486 Tanger Hamid Amina 40 842 Le Caire Zahir Khalid 54 1013 Riyad Qasimi Omar 49 790 Tunis Cherkaoui Ahmed 55 860 Tunis Saadi Leila 24 894 Tanger Qasimi Ahmed 40 399 Casablanca Alami Mohamed 43 267 Riyad El Fassi Ahmed 36 438 Fès Benjelloun Amina 24 214 Tanger Mansour Khalid 57 806 Tanger Qasimi Omar 53 333 Alexandrie Qasimi Fatima 30 745 Alexandrie El Fassi Mohamed 31 121 Le Caire Saadi Fatima 25 766 Tanger Qasimi Leila 32 846 Le Caire Bouazza Amina 28 115 Riyad Bouazza Mohamed 42 979 Marrakech Hamid Hafsa 35 546 Alexandrie Oualid Ahmed 37 895 Dubaï Saadi Youssef 47 263 Casablanca Idrissi Fatima 25 118 Fès Alami Youssef 37 536 Dubaï Hiba Zahra 57 575 Fès Zahir Ahmed 27 507 Alexandrie Zahir Zahra 30 217 Alexandrie Hamid Fatima 60 545 Tunis Bouazza Ahmed 45 133 Marrakech Mansour Omar 45 1043 Tunis Neha Amina 20 505 Fès Cherkaoui Omar 42 472 Rabat Hamid Zahra 36 271 Fès Benjelloun Mohamed 60 189 Alexandrie Oualid Omar 59 791 Alexandrie Alami Mohamed 44 904 Casablanca Bouazza Hafsa 25 785 Fès Mansour Khalid 35 996 Tanger Neha Fatima 44 289 Tanger Idrissi Hafsa 59 798 Tanger Bouazza Leila 28 1024 Fès Qasimi Ahmed 44 1052 Casablanca El Fassi Ahmed 35 132 Fès Alami Ahmed 52 816 Casablanca Bouazza Zahra 51 361 Tanger Saadi Ahmed 48 377 Rabat Alami Mohamed 27 407 Casablanca Saadi Ahmed 50 823 Rabat Idrissi Youssef 34 643 Casablanca Cherkaoui Youssef 41 344 Tanger Oualid Fatima 47 592 Rabat El Fassi Amina 33 831 Marrakech Oualid Ahmed 27 322 Marrakech Zahir Omar 60 735 Tanger Oualid Amina 27 937 Fès Alami Amina 30 663 Rabat Cherkaoui Omar 22 922 Le Caire Zahir Zahra 53 713 Le Caire Saadi Ahmed 29 668 Dubaï Bouazza Mohamed 32 731 Tanger Hamid Youssef 48 569 Tunis Mansour Leila 24 515 Rabat Cherkaoui Khalid 22 687 Dubaï Alami Mohamed 46 574 Riyad Cherkaoui Zahra 27 112 Marrakech Bouazza Mohamed 52 237 Tanger Hiba Khalid 45 139 Rabat Bouazza Omar 56 1049 Riyad Neha Amina 33 191 Marrakech Alami Hafsa 55 653 Dubaï Mansour Khalid 55 992 Dubaï Saadi Mohamed 39 617 Alexandrie Neha Hafsa 37 834 Alexandrie Bouazza Youssef 48 916 Rabat Saadi Hafsa 28 675 Tanger Neha Hafsa 59 803 Rabat Benjelloun Hafsa 57 209 Rabat Saadi Zahra 50 445 Riyad Qasimi Leila 56 758 Riyad Hamid Leila 60 100 Marrakech Hiba Ahmed 60 702 Marrakech Neha Khalid 31 142 Casablanca Zahir Amina 29 510 Tanger Neha Leila 41 727 Tanger Saadi Amina 52 457 Casablanca Benjelloun Hafsa 28 670 Tunis Benjelloun Mohamed 28 630 Marrakech Hiba Zahra 32 218 Le Caire Neha Zahra 25 1069 Rabat Idrissi Omar 50 512 Riyad Oualid Amina 32 925 Dubaï Hiba Fatima 48 146 Tanger Hiba Leila 27 287 Fès Benjelloun Ahmed 20 781 Casablanca Cherkaoui Khalid 43 974 Rabat Zahir Khalid 35 724 Tunis Alami Omar 46 625 Tanger Hamid Hafsa 40 200 Dubaï Bouazza Leila 48 273 Marrakech Idrissi Mohamed 24 869 Tunis Hamid Amina 24 801 Dubaï Hamid Hafsa 26 917 Dubaï Saadi Fatima 25 631 Marrakech Alami Leila 38 123 Tanger Benjelloun Khalid 57 1087 Dubaï Saadi Youssef 24 342 Casablanca Hiba Mohamed 20 942 Tanger Bouazza Khalid 52 720 Casablanca Neha Fatima 35 883 Riyad Hamid Omar 39 760 Tunis Zahir Omar 40 805 Dubaï Saadi Zahra 60 467 Tunis Oualid Leila 28 415 Alexandrie Benjelloun Hafsa 37 570 Tanger Alami Omar 20 1062 Tunis Cherkaoui Amina 31 396 Riyad Oualid Leila 20 230 Tanger Neha Leila 59 504 Riyad Qasimi Omar 31 162 Rabat Zahir Leila 45 542 Le Caire Alami Youssef 58 130 Le Caire Mansour Hafsa 45 577 Dubaï Oualid Ahmed 36 830 Dubaï Idrissi Zahra 28 516 Fès Mansour Mohamed 56 676 Rabat Qasimi Youssef 42 603 Le Caire Alami Amina 58 855 Riyad Cherkaoui Mohamed 60 755 Riyad Qasimi Amina 51 222 Tanger Zahir Omar 48 313 Riyad Alami Ahmed 54 911 Tanger Zahir Youssef 32 941 Fès Qasimi Youssef 44 955 Marrakech Idrissi Zahra 26 354 Alexandrie El Fassi Zahra 34 580 Tanger Hamid Hafsa 44 179 Casablanca Alami Omar 44 786 Le Caire El Fassi Mohamed 51 993 Rabat Neha Ahmed 29 1000 Alexandrie El Fassi Youssef 54 618 Fès Alami Amina 46 747 Marrakech Benjelloun Ahmed 59 780 Alexandrie Oualid Zahra 39 427 Tanger Zahir Omar 23 395 Rabat Oualid Mohamed 23 155 Tunis Oualid Ahmed 42 964 Marrakech Neha Ahmed 55 708 Tunis Hamid Zahra 29 330 Fès Mansour Mohamed 54 314 Casablanca Cherkaoui Youssef 47 318 Fès El Fassi Youssef 38 865 Le Caire Mansour Zahra 36 797 Dubaï Mansour Amina 29 915 Le Caire Benjelloun Ahmed 29 245 Marrakech Zahir Khalid 50 596 Marrakech El Fassi Zahra 30 173 Dubaï Oualid Fatima 41 1033 Tunis Idrissi Hafsa 46 409 Le Caire Alami Youssef 48 153 Marrakech Benjelloun Amina 27 116 Riyad Bouazza Leila 49 274 Tunis Zahir Omar 54 931 Marrakech Oualid Amina 37 1072 Fès Saadi Hafsa 24 442 Fès Benjelloun Khalid 32 919 Tunis Qasimi Khalid 49 556 Dubaï Neha Fatima 49 114 Rabat Qasimi Omar 28 349 Le Caire Qasimi Youssef 40 345 Rabat Saadi Fatima 27 400 Alexandrie Hamid Leila 60 954 Rabat Saadi Zahra 39 171 Rabat Qasimi Youssef 21 584 Dubaï Hiba Ahmed 36 939 Rabat Mansour Hafsa 35 532 Alexandrie Hiba Amina 60 560 Fès Saadi Khalid 38 166 Dubaï Saadi Hafsa 23 887 Riyad Hamid Youssef 56 723 Casablanca Mansour Fatima 36 934 Marrakech Idrissi Ahmed 40 335 Marrakech El Fassi Ahmed 44 302 Tanger Oualid Omar 60 109 Riyad Saadi Khalid 48 819 Tunis Hamid Youssef 24 770 Le Caire Cherkaoui Ahmed 46 826 Rabat Hamid Amina 48 658 Tunis Zahir Leila 41 362 Le Caire El Fassi Hafsa 35 1019 Tanger Idrissi Amina 33 439 Le Caire Benjelloun Leila 26 880 Fès Benjelloun Mohamed 48 1035 Le Caire Hamid Khalid 33 529 Marrakech Zahir Youssef 29 748 Alexandrie Zahir Youssef 22 642 Dubaï Bouazza Amina 41 1005 Marrakech Idrissi Amina 37 485 Fès Alami Youssef 52 283 Marrakech El Fassi Mohamed 41 315 Le Caire Mansour Youssef 23 614 Fès Benjelloun Amina 22 494 Tunis Saadi Ahmed 58 686 Le Caire Hamid Mohamed 29 432 Casablanca Zahir Fatima 36 151 Riyad Saadi Youssef 44 468 Marrakech Saadi Hafsa 43 655 Casablanca Neha Youssef 42 408 Fès Alami Ahmed 26 343 Riyad Oualid Mohamed 49 962 Casablanca Hamid Zahra 48 355 Casablanca Saadi Fatima 28 497 Alexandrie El Fassi Omar 37 893 Marrakech Hiba Ahmed 20 397 Alexandrie Qasimi Youssef 25 850 Alexandrie Alami Mohamed 24 890 Riyad Oualid Omar 47 125 Alexandrie Idrissi Youssef 30 402 Tanger Idrissi Ahmed 49 319 Riyad Benjelloun Omar 39 480 Alexandrie Benjelloun Hafsa 45 316 Fès Hamid Omar 20 606 Le Caire Zahir Youssef 48 1038 Alexandrie Saadi Mohamed 35 980 Riyad Qasimi Leila 57 360 Dubaï Saadi Hafsa 56 549 Fès Zahir Mohamed 21 364 Le Caire Zahir Zahra 40 462 Marrakech Mansour Mohamed 37 1016 Alexandrie Idrissi Mohamed 37 426 Rabat Hiba Mohamed 54 693 Tunis Cherkaoui Fatima 45 444 Dubaï Alami Amina 23 220 Fès Mansour Omar 27 807 Marrakech Bouazza Omar 60 629 Le Caire Hiba Zahra 26 1039 Marrakech Hamid Mohamed 50 352 Fès Hiba Youssef 47 1090 Fès Hamid Youssef 46 749 Casablanca Saadi Khalid 32 381 Marrakech Zahir Mohamed 44 1028 Tanger Cherkaoui Fatima 27 428 Le Caire Cherkaoui Fatima 45 483 Fès Idrissi Amina 25 524 Dubaï Bouazza Youssef 38 339 Fès voici le contenu nom prenon age nm de compte ville

**ChatGPT a dit :**

Il semble que vous ayez partagé une liste de données sous forme de tableau avec des informations sur des personnes, telles que leur nom, prénom, âge, numéro de compte et ville. Que souhaitez-vous faire avec ces données ? Voulez-vous les insérer dans une base de données, effectuer une analyse spécifique, ou les utiliser d'une autre manière ?

**Vous avez dit :**

c'est un fichier plat

**ChatGPT a dit :**

D'accord, donc vous avez un fichier plat qui contient des informations sur des personnes (nom, prénom, âge, numéro de compte, ville) et vous voulez peut-être le traiter dans un format particulier, ou peut-être l'analyser ou le convertir dans une autre structure (comme un fichier CSV, une base de données, etc.).

Si vous souhaitez manipuler ce fichier, il y a plusieurs étapes que l'on peut suivre. Voici quelques questions pour mieux comprendre ce que vous souhaitez faire :

1. **Format du fichier plat** : Est-ce que chaque ligne est une entrée unique avec les informations séparées par des espaces, des tabulations, ou un autre caractère spécifique ?
2. **Conversion** : Voulez-vous convertir ce fichier plat en un autre format, comme un fichier CSV ou JSON ?
3. **Analyse** : Souhaitez-vous effectuer une analyse spécifique sur ces données (par exemple, trier les personnes par âge, rechercher par ville, etc.) ?

Si vous avez une idée plus précise de ce que vous souhaitez faire, je peux vous guider dans la manipulation et le traitement de ces données.

**Vous avez dit :**

chaque ligne contient le nom et prenom , age , nm de compte , et la ville sans separtion juste en jouant sur la position 0\_15 nom 16\_30 prenom 31\_33 age 34\_43 nm de compte 44 \_63 ville

**ChatGPT a dit :**

Merci pour la précision. Donc chaque ligne du fichier plat contient les informations suivantes avec une séparation basée sur la position :

* **Position 0 à 15** : Nom
* **Position 16 à 30** : Prénom
* **Position 31 à 33** : Âge
* **Position 34 à 43** : Numéro de compte
* **Position 44 à 63** : Ville

Voici un exemple de traitement possible avec ce type de format.

**Objectif**

L'objectif ici pourrait être de lire le fichier, d'extraire les informations selon les positions mentionnées et de les enregistrer sous un format structuré, comme un fichier CSV ou une base de données.

**Solution**

Je vais vous proposer un code Python pour extraire ces informations ligne par ligne, puis les enregistrer dans un fichier CSV.

**Code Python pour traiter ce fichier plat**

python

CopierModifier

import csv def process\_line(line): *# Extraire les informations en fonction des positions* nom = line[0:16].strip() *# Position 0-15 pour le nom* prenom = line[16:31].strip() *# Position 16-30 pour le prénom* age = line[31:34].strip() *# Position 31-33 pour l'âge* num\_compte = line[34:44].strip() *# Position 34-43 pour le numéro de compte* ville = line[44:64].strip() *# Position 44-63 pour la ville* return [nom, prenom, age, num\_compte, ville] def process\_file(input\_filename, output\_filename): with open(input\_filename, 'r', encoding='utf-8') as infile, open(output\_filename, 'w', newline='', encoding='utf-8') as outfile: *# Création du writer pour écrire les données dans le fichier CSV* csv\_writer = csv.writer(outfile) *# Écrire l'en-tête du CSV* csv\_writer.writerow(["Nom", "Prénom", "Âge", "Numéro de compte", "Ville"]) *# Lire chaque ligne du fichier* for line in infile: *# S'assurer qu'il ne s'agit pas d'une ligne vide* if line.strip(): *# Traiter la ligne et extraire les données* data = process\_line(line) *# Écrire les données dans le fichier CSV* csv\_writer.writerow(data) *# Utilisation des fonctions* input\_filename = 'data.txt' *# Remplacer par le nom de votre fichier d'entrée* output\_filename = 'output.csv' *# Fichier CSV de sortie* process\_file(input\_filename, output\_filename)

**Explication du code**

1. **process\_line(line)** :
   * Cette fonction prend une ligne du fichier plat, extrait les différentes informations basées sur les positions définies (comme vous l'avez décrit), puis renvoie une liste avec ces informations.
2. **process\_file(input\_filename, output\_filename)** :
   * Ouvre le fichier d'entrée (data.txt), lit chaque ligne, et appelle la fonction process\_line() pour extraire les informations de chaque ligne.
   * Ensuite, elle écrit ces informations dans un fichier CSV (output.csv), avec une ligne d'en-tête.
3. **CSV Writer** :
   * Le module `csv

**Vous avez dit :**

je veux que change le projet pour matcher le format de ce fichier plat

**ChatGPT a dit :**

Pour adapter le projet au format de votre fichier plat, où les informations sont extraites selon les positions que vous avez spécifiées, il faut faire en sorte que le script :

* Lise chaque ligne du fichier et extraie les informations selon les indices des caractères que vous avez fournis.
* Enregistre ces données dans un fichier structuré, comme un fichier CSV ou une base de données.

**Format de votre fichier plat**

D'après votre description, chaque ligne contient les données dans les positions suivantes :

1. **Nom** : Position 0-15
2. **Prénom** : Position 16-30
3. **Âge** : Position 31-33
4. **Numéro de compte** : Position 34-43
5. **Ville** : Position 44-63

**Code Python adapté au fichier plat**

Voici le code mis à jour pour extraire ces informations en fonction des positions et les enregistrer dans un fichier CSV.

**Code Python**

python

CopierModifier

import csv *# Fonction pour extraire les informations en fonction des indices de position* def process\_line(line): *# Extraire les informations en fonction des positions* nom = line[0:16].strip() *# Position 0-15 pour le nom* prenom = line[16:31].strip() *# Position 16-30 pour le prénom* age = line[31:34].strip() *# Position 31-33 pour l'âge* num\_compte = line[34:44].strip() *# Position 34-43 pour le numéro de compte* ville = line[44:64].strip() *# Position 44-63 pour la ville* return [nom, prenom, age, num\_compte, ville] *# Fonction principale pour lire le fichier et écrire dans un fichier CSV* def process\_file(input\_filename, output\_filename): with open(input\_filename, 'r', encoding='utf-8') as infile, open(output\_filename, 'w', newline='', encoding='utf-8') as outfile: *# Création du writer pour écrire les données dans le fichier CSV* csv\_writer = csv.writer(outfile) *# Écrire l'en-tête du CSV* csv\_writer.writerow(["Nom", "Prénom", "Âge", "Numéro de compte", "Ville"]) *# Lire chaque ligne du fichier* for line in infile: *# S'assurer qu'il ne s'agit pas d'une ligne vide* if line.strip(): *# Traiter la ligne et extraire les données* data = process\_line(line) *# Écrire les données dans le fichier CSV* csv\_writer.writerow(data) *# Utilisation des fonctions* input\_filename = 'data.txt' *# Remplacer par le nom de votre fichier d'entrée* output\_filename = 'output.csv' *# Fichier CSV de sortie* process\_file(input\_filename, output\_filename)

**Explication du code**

1. process\_line(line) :
   * Cette fonction reçoit une ligne brute du fichier plat.
   * Elle extrait les différentes informations (nom, prénom, âge, numéro de compte, ville) en utilisant les indices de caractères que vous avez donnés.
   * La fonction strip() est utilisée pour enlever les espaces ou caractères de fin inutiles.
2. process\_file(input\_filename, output\_filename) :
   * Ouvre le fichier d'entrée (data.txt) en mode lecture et le fichier de sortie (output.csv) en mode écriture.
   * Chaque ligne du fichier d'entrée est lue et traitée via la fonction process\_line().
   * Ensuite, les informations extraites sont écrites dans un fichier CSV avec

**Vous avez dit :**

le projet est celui de traitement sur le quel on parlait avant

**ChatGPT a dit :**

Ah, je comprends maintenant ! Vous souhaitez adapter le projet de traitement des fichiers plats (avec les informations sur les noms, prénoms, âges, etc.) à l'ancien projet que nous avions discuté. Cela signifie que vous souhaitez toujours traiter ces données selon les positions des caractères, mais avec l'intégration des fonctionnalités supplémentaires que nous avions abordées dans le projet initial.

**Objectif :**

* Lire un fichier plat où les informations sont positionnées de manière fixe, comme vous l'avez décrit.
* Ajouter des fonctionnalités de traitement, comme la validation des données, l'exportation dans un fichier structuré, et peut-être des calculs ou analyses supplémentaires sur les informations extraites (par exemple, filtrer les utilisateurs en fonction de l'âge ou de la ville).

**Voici l’adaptation du projet complet avec ces détails :**

1. **Lire et extraire les informations à partir du fichier plat.**
2. **Analyser et valider les données.**
3. **Exporter les données dans un format structuré (par exemple CSV).**

Je vais réécrire le code avec l'intégration de la validation et de l'exportation.

**Nouveau Code Complet pour le Projet**

**1. Lecture et Extraction des Données**

Le code suivant traite le fichier plat avec les informations aux positions définies, les valide, puis les exporte sous forme de fichier CSV.

python

CopierModifier

import csv *# Fonction pour extraire les informations en fonction des positions des caractères* def process\_line(line): *# Extraire les informations en fonction des positions* nom = line[0:16].strip() *# Position 0-15 pour le nom* prenom = line[16:31].strip() *# Position 16-30 pour le prénom* age = line[31:34].strip() *# Position 31-33 pour l'âge* num\_compte = line[34:44].strip() *# Position 34-43 pour le numéro de compte* ville = line[44:64].strip() *# Position 44-63 pour la ville* *# Retourner les données extraites sous forme de liste* return [nom, prenom, age, num\_compte, ville] *# Validation des données* def validate\_data(nom, prenom, age, num\_compte, ville): *# Vérification du nom et du prénom (doivent être non vides)* if not nom or not prenom: return False, "Nom ou prénom manquant" *# Vérification de l'âge (doit être un nombre et dans une plage raisonnable)* try: age = int(age) if age < 18 or age > 120: return False, "Âge non valide" except ValueError: return False, "Âge non valide" *# Vérification du numéro de compte (doit être composé de chiffres)* if not num\_compte.isdigit(): return False, "Numéro de compte invalide" *# Vérification de la ville (doit être non vide)* if not ville: return False, "Ville manquante" return True, "" *# Fonction principale pour traiter le fichier* def process\_file(input\_filename, output\_filename): with open(input\_filename, 'r', encoding='utf-8') as infile, open(output\_filename, 'w', newline='', encoding='utf-8') as outfile: *# Création du writer pour écrire les données dans le fichier CSV* csv\_writer = csv.writer(outfile) *# Écrire l'en-tête du CSV* csv\_writer.writerow(["Nom", "Prénom", "Âge", "Numéro de compte", "Ville", "Validité"]) *# Lire chaque ligne du fichier* for line in infile: *# S'assurer qu'il ne s'agit pas d'une ligne vide* if line.strip(): *# Traiter la ligne et extraire les données* data = process\_line(line) *# Valider les données extraites* valid, message = validate\_data(\*data) *# Ajouter un champ de validation dans les données* data.append('Valide' if valid else f'Invalide: {message}') *# Écrire les données (avec validation) dans le fichier CSV* csv\_writer.writerow(data) *# Utilisation des fonctions* input\_filename = 'data.txt' *# Remplacer par le nom de votre fichier d'entrée* output\_filename = 'output.csv' *# Fichier CSV de sortie* process\_file(input\_filename, output\_filename)

**Explication des nouvelles fonctionnalités**

1. process\_line(line) :
   * Fonction qui extrait les informations de chaque ligne du fichier plat en utilisant les indices de position spécifiés (nom, prénom, âge, numéro de compte, ville).
2. validate\_data(nom, prenom, age, num\_compte, ville) :
   * Fonction qui valide les données extraites. Elle vérifie :
     + Si le **nom** et le **prénom** sont non vides.
     + Si l'**âge** est un nombre valide et dans une plage réaliste (entre 18 et 120 ans).
     + Si le **numéro de compte** est composé uniquement de chiffres.
     + Si la **ville** est non vide.

Si les données sont valides, la fonction retourne True et une chaîne vide. Si elles sont invalides, elle retourne False avec un message d'erreur spécifique.

1. process\_file(input\_filename, output\_filename) :
   * Cette fonction gère la lecture du fichier d'entrée, le traitement de chaque ligne, la validation des données et l'écriture des résultats dans un fichier CSV.
   * Le fichier CSV contient maintenant une colonne supplémentaire **"Validité"**, qui indique si les données sont valides ou non.

**Exemple d'Entrée et Sortie**

**Exemple d'entrée (fichier**data.txt**):**

yaml

CopierModifier

Neha Amina 51 1029 Casablanca

Qasimi Youssef 58 548 Tanger

Hamid Hafsa 25 370 Fès

Neha Khalid 52 1095 Casablanca

**Exemple de sortie (fichier**output.csv**):**

r

CopierModifier

Nom,Prénom,Âge,Numéro de compte,Ville,Validité

Neha,Amina,51,1029,Casablanca,Valide

Qasimi,Youssef,58,548,Tanger,Valide

Hamid,Hafsa,25,370,Fès,Valide

Neha,Khalid,52,1095,Casablanca,Valide

**Améliorations possibles**

1. **Gestion des erreurs** : Ajouter des logs pour les erreurs qui pourraient survenir pendant la lecture ou l'écriture des fichiers.
2. **Traitements supplémentaires** : Par exemple, filtrer ou trier les résultats par âge ou ville, ou encore effectuer des analyses statistiques sur les données extraites.

Si vous avez des exigences supplémentaires ou des spécifications particulières, je peux adapter le code en conséquence.

**Vous avez dit :**

c'est un projet springbatch

**ChatGPT a dit :**

Ah, d'accord ! Vous travaillez avec **Spring Batch** pour traiter le fichier plat et le transformer en une sortie structurée (comme un fichier CSV). Je vais donc vous donner une solution basée sur Spring Batch pour traiter le fichier plat avec des informations en fonction de positions fixes, valider les données et exporter les résultats.

**Objectif :**

1. Lire un fichier plat où les informations sont positionnées de manière fixe (nom, prénom, âge, numéro de compte, ville).
2. Traiter chaque ligne du fichier, valider les données et écrire le résultat dans un fichier CSV.
3. Utiliser Spring Batch pour gérer le traitement par lots de manière scalable et robuste.

**Étapes :**

1. **Configurer Spring Batch avec un job et des étapes.**
2. **Créer un**ItemReader**pour lire les données du fichier plat.**
3. **Créer un**ItemProcessor**pour valider les données.**
4. **Créer un**ItemWriter**pour écrire les données validées dans un fichier CSV.**

**1. Dépendances Maven**

Tout d'abord, vous devez inclure les dépendances nécessaires dans votre fichier pom.xml pour Spring Batch et Spring Boot.

xml

CopierModifier

<dependencies> *<!-- Spring Boot Starter pour le batch processing -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> *<!-- Spring Batch core -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> </dependency> *<!-- Spring Boot Starter pour l'accès aux fichiers CSV -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-infrastructure</artifactId> </dependency> *<!-- Spring Boot Starter Web pour la gestion des services REST (facultatif) -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency> *<!-- CSV Support: Apache Commons CSV -->* <dependency> <groupId>org.apache.commons</groupId> <artifactId>commons-csv</artifactId> <version>1.8</version> </dependency> </dependencies>

**2. Configuration de Spring Batch**

Créons une configuration Spring Batch dans un fichier de configuration Java. Ce fichier va définir le job, les étapes, et les composants de lecture, traitement, et écriture.

BatchConfig.java

java

CopierModifier

import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.repository.JobRepository; import org.springframework.batch.core.repository.support.MapJobRepositoryFactoryBean; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemWriter; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.FixedLengthTokenizer; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; import org.springframework.batch.core.Job; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.core.io.FileSystemResource; import org.springframework.batch.item.file.transform.DefaultLineMapper; import org.springframework.batch.item.file.transform.LineTokenizer; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job job(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, ItemReader<Person> reader, ItemProcessor<Person, Person> processor, ItemWriter<Person> writer) { Step step = stepBuilderFactory.get("step1") .<Person, Person> chunk(10) .reader(reader) .processor(processor) .writer(writer) .build(); return jobBuilderFactory.get("job1") .incrementer(new RunIdIncrementer()) .start(step) .build(); } @Bean public FlatFileItemReader<Person> reader() { FlatFileItemReader<Person> reader = new FlatFileItemReader<>(); reader.setResource(new FileSystemResource("data.txt")); *// Fichier d'entrée* reader.setLineMapper(new DefaultLineMapper<Person>() {{ setLineTokenizer(new FixedLengthTokenizer() {{ setColumns(new Range(0, 15), new Range(16, 30), new Range(31, 33), new Range(34, 43), new Range(44, 63)); }}); setFieldSetMapper(new PersonFieldSetMapper()); }}); return reader; } @Bean public ItemProcessor<Person, Person> processor() { return new PersonItemProcessor(); } @Bean public ItemWriter<Person> writer() { return new PersonItemWriter(); } }

**3. Définir le**ItemReader

L'ItemReader va lire chaque ligne du fichier plat en fonction des positions des colonnes (en utilisant des Range comme vous l'avez indiqué). Chaque ligne sera mappée à un objet Person.

PersonFieldSetMapper.java

java

CopierModifier

import org.springframework.batch.item.file.mapping.FieldSetMapper; import org.springframework.batch.item.file.transform.FieldSet; public class PersonFieldSetMapper implements FieldSetMapper<Person> { @Override public Person mapFieldSet(FieldSet fieldSet) { String nom = fieldSet.readString(0); String prenom = fieldSet.readString(1); int age = fieldSet.readInt(2); String numCompte = fieldSet.readString(3); String ville = fieldSet.readString(4); return new Person(nom, prenom, age, numCompte, ville); } }

Person.java

java

CopierModifier

public class Person { private String nom; private String prenom; private int age; private String numCompte; private String ville; *// Constructeurs, getters et setters* public Person(String nom, String prenom, int age, String numCompte, String ville) { this.nom = nom; this.prenom = prenom; this.age = age; this.numCompte = numCompte; this.ville = ville; } *// Getters et Setters* }

**4. Définir le**ItemProcessor

L'ItemProcessor va valider les données extraites de chaque ligne et renvoyer l'objet Person seulement si les données sont valides.

PersonItemProcessor.java

java

CopierModifier

public class PersonItemProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) throws Exception { if (item.getNom() == null || item.getPrenom() == null || item.getAge() < 18 || item.getNumCompte().length() != 10) { return null; *// Retourne null si l'objet est invalide* } return item; *// Retourne l'objet validé* } }

**5. Définir le**ItemWriter

L'ItemWriter va écrire les objets Person dans un fichier CSV.

PersonItemWriter.java

java

CopierModifier

import org.springframework.batch.item.file.FlatFileItemWriter; import org.springframework.batch.item.file.mapping.BeanWrapperFieldExtractor; import org.springframework.batch.item.file.transform.DelimitedLineAggregator; import org.springframework.batch.item.file.transform.LineAggregator; import org.springframework.batch.item.file.transform.FieldExtractor; public class PersonItemWriter extends FlatFileItemWriter<Person> { public PersonItemWriter() { setResource(new FileSystemResource("output.csv")); setAppendAllowed(false); *// Configuration du LineAggregator* DelimitedLineAggregator<Person> lineAggregator = new DelimitedLineAggregator<>(); lineAggregator.setDelimiter(","); BeanWrapperFieldExtractor<Person> fieldExtractor = new BeanWrapperFieldExtractor<>(); fieldExtractor.setNames(new String[]{"nom", "prenom", "age", "numCompte", "ville"}); lineAggregator.setFieldExtractor(fieldExtractor); setLineAggregator(lineAggregator); } }

**6. Exécution du Job**

Le JobLauncher est utilisé pour lancer le job Spring Batch. Si vous utilisez un serveur d'applications ou une application Spring Boot, vous pouvez lancer le job via un contrôleur ou directement dans la classe principale.

**Exemple de Classe Principale (**SpringBatchApplication.java**)**

java

CopierModifier

import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.Job; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.boot.CommandLineRunner; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class SpringBatchApplication implements CommandLineRunner { @Autowired private JobLauncher jobLauncher; @Autowired private Job job; public static void main(String[] args) { SpringApplication.run(SpringBatchApplication.class, args); } @Override public void run(String... args) throws Exception { jobLauncher.run(job, new JobParameters()); } }

**7. Lancer le Job**

Après avoir configuré Spring Batch, il suffit de lancer l'application pour démarrer le traitement du fichier plat et générer un fichier CSV valide. Le processus traitera chaque ligne du fichier plat, validera les données et les écrira dans un fichier CSV.

Ce projet Spring Batch vous permet de traiter efficacement des fichiers plats tout en validant et en exportant les données. Si vous avez des questions supplémentaires ou besoin de plus de précisions, n'hésitez pas à me le faire savoir !

**Vous avez dit :**

src ├── main │ ├── java │ │ └── com │ │ └── exemple │ │ └── traitement │ │ ├── config │ │ │ └── BatchConfig.java │ │ ├── model │ │ │ └── Person.java │ │ ├── processor │ │ │ └── PersonProcessor.java │ │ ├── reader │ │ │ └── PersonReader.java │ │ ├── writer │ │ │ └── PersonWriter.java │ │ └── TraitementApplication.java │ ├── resources │ │ └── input │ │ └── people.txt │ │ └── output │ │ └── result.json │ ├── application.properties ├── pom.xml

**ChatGPT a dit :**

Voici un aperçu de la structure de votre projet Spring Batch tel que vous l'avez décrit, avec des ajustements et des recommandations pour chaque composant. L'organisation du projet est correcte et suit une architecture typique pour un projet Spring Batch.

Voici une vue d'ensemble des fichiers à créer et des recommandations pour chaque répertoire et classe de votre projet.

**1. Structure du projet**

css

CopierModifier

src

├── main

│ ├── java

│ │ └── com

│ │ └── exemple

│ │ └── traitement

│ │ ├── config

│ │ │ └── BatchConfig.java

│ │ ├── model

│ │ │ └── Person.java

│ │ ├── processor

│ │ │ └── PersonProcessor.java

│ │ ├── reader

│ │ │ └── PersonReader.java

│ │ ├── writer

│ │ │ └── PersonWriter.java

│ │ └── TraitementApplication.java

│ ├── resources

│ │ └── input

│ │ └── people.txt

│ │ └── output

│ │ └── result.csv

│ ├── application.properties

├── pom.xml

**2. Détails des classes et fichiers**

**a.**BatchConfig.java

Cette classe est la configuration de Spring Batch, où nous définissons le job et les étapes (reader, processor, writer).

java

CopierModifier

package com.exemple.traitement.config; import com.exemple.traitement.model.Person; import com.exemple.traitement.processor.PersonProcessor; import com.exemple.traitement.reader.PersonReader; import com.exemple.traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.repository.JobRepository; import org.springframework.batch.core.repository.support.MapJobRepositoryFactoryBean; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemWriter; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.FixedLengthTokenizer; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; import org.springframework.batch.core.Job; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.core.io.FileSystemResource; import org.springframework.batch.item.file.transform.DefaultLineMapper; import org.springframework.batch.item.file.transform.LineTokenizer; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job job(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, ItemReader<Person> reader, ItemProcessor<Person, Person> processor, ItemWriter<Person> writer) { Step step = stepBuilderFactory.get("step1") .<Person, Person> chunk(10) .reader(reader) .processor(processor) .writer(writer) .build(); return jobBuilderFactory.get("job1") .incrementer(new RunIdIncrementer()) .start(step) .build(); } @Bean public ItemReader<Person> reader() { return new PersonReader(); } @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); } @Bean public ItemWriter<Person> writer() { return new PersonWriter(); } }

**b.**Person.java

Le modèle de données représentant une personne, avec des attributs pour le nom, le prénom, l'âge, le numéro de compte et la ville.

java

CopierModifier

package com.exemple.traitement.model; public class Person { private String nom; private String prenom; private int age; private String numCompte; private String ville; *// Constructeurs, getters et setters* public Person(String nom, String prenom, int age, String numCompte, String ville) { this.nom = nom; this.prenom = prenom; this.age = age; this.numCompte = numCompte; this.ville = ville; } public String getNom() { return nom; } public void setNom(String nom) { this.nom = nom; } public String getPrenom() { return prenom; } public void setPrenom(String prenom) { this.prenom = prenom; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getNumCompte() { return numCompte; } public void setNumCompte(String numCompte) { this.numCompte = numCompte; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } }

**c.**PersonProcessor.java

Le processeur qui valide les données. Il peut être amélioré pour filtrer ou transformer les données si nécessaire.

java

CopierModifier

package com.exemple.traitement.processor; import com.exemple.traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) throws Exception { *// Validation des données (exemple basique)* if (item.getNom() == null || item.getPrenom() == null || item.getAge() < 18 || item.getNumCompte().length() != 10) { return null; *// Retourne null si l'objet est invalide* } return item; *// Retourne l'objet validé* } }

**d.**PersonReader.java

Le lecteur qui extrait les données du fichier plat en fonction des positions de colonnes spécifiées.

java

CopierModifier

package com.exemple.traitement.reader; import com.exemple.traitement.model.Person; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.FixedLengthTokenizer; import org.springframework.batch.item.file.transform.DefaultLineMapper; import org.springframework.batch.item.file.transform.LineTokenizer; import org.springframework.core.io.FileSystemResource; public class PersonReader extends FlatFileItemReader<Person> { public PersonReader() { setResource(new FileSystemResource("src/main/resources/input/people.txt")); setLineMapper(new DefaultLineMapper<Person>() {{ setLineTokenizer(new FixedLengthTokenizer() {{ setColumns(new Range(0, 15), new Range(16, 30), new Range(31, 33), new Range(34, 43), new Range(44, 63)); }}); setFieldSetMapper(new PersonFieldSetMapper()); }}); } }

**e.**PersonWriter.java

L'écrivain qui écrit les données traitées dans un fichier CSV.

java

CopierModifier

package com.exemple.traitement.writer; import com.exemple.traitement.model.Person; import org.springframework.batch.item.file.FlatFileItemWriter; import org.springframework.batch.item.file.mapping.BeanWrapperFieldExtractor; import org.springframework.batch.item.file.transform.DelimitedLineAggregator; import org.springframework.batch.item.file.transform.LineAggregator; public class PersonWriter extends FlatFileItemWriter<Person> { public PersonWriter() { setResource(new FileSystemResource("src/main/resources/output/result.csv")); setAppendAllowed(false); *// Configuration du LineAggregator* DelimitedLineAggregator<Person> lineAggregator = new DelimitedLineAggregator<>(); lineAggregator.setDelimiter(","); BeanWrapperFieldExtractor<Person> fieldExtractor = new BeanWrapperFieldExtractor<>(); fieldExtractor.setNames(new String[]{"nom", "prenom", "age", "numCompte", "ville"}); lineAggregator.setFieldExtractor(fieldExtractor); setLineAggregator(lineAggregator); } }

**f.**TraitementApplication.java

La classe principale de l'application Spring Boot qui lance le job Spring Batch.

java

CopierModifier

package com.exemple.traitement; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.Job; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.boot.CommandLineRunner; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class TraitementApplication implements CommandLineRunner { @Autowired private JobLauncher jobLauncher; @Autowired private Job job; public static void main(String[] args) { SpringApplication.run(TraitementApplication.class, args); } @Override public void run(String... args) throws Exception { jobLauncher.run(job, new JobParameters()); } }

**3. Fichiers d'entrée et de sortie**

**a.**people.txt**(Fichier d'entrée)**

Ce fichier doit contenir des lignes formatées selon les positions fixes, par exemple :

nginx

CopierModifier

John Doe 25 1234567890 Paris

Jane Smith 32 2345678901 Lyon

**b.**result.csv**(Fichier de sortie)**

Le fichier de sortie sera généré par le ItemWriter, avec un format CSV :

CopierModifier

nom,prenom,age,numCompte,ville

John,Doe,25,1234567890,Paris

Jane,Smith,32,2345678901,Lyon

**4. Configuration de**application.properties

Si vous avez besoin de configurer des paramètres spécifiques à Spring Batch, vous pouvez utiliser le fichier application.properties.

properties

CopierModifier

spring.batch.job.enabled=true spring.batch.initialize-schema=always

Cela devrait couvrir les bases de votre projet Spring Batch pour traiter des fichiers plats et générer des fichiers CSV. Si vous avez d'autres questions ou des spécifications supplémentaires, n'hésitez pas à demander !

**Vous avez dit :**

je veux un fichier json a la fin pas csv

**ChatGPT a dit :**

D'accord, si vous souhaitez générer un fichier JSON au lieu d'un fichier CSV à la fin du processus, nous devons ajuster l'écrivain (ItemWriter) pour qu'il écrive les données sous forme de JSON. Nous allons utiliser une bibliothèque comme Jackson pour sérialiser les objets Person en JSON.

Voici les étapes pour ajuster le projet afin qu'il écrive un fichier JSON au lieu d'un fichier CSV.

**1. Modifier l'écrivain (**PersonWriter.java**)**

Nous allons utiliser ObjectMapper de Jackson pour convertir les objets Person en JSON et les écrire dans un fichier.

java

CopierModifier

package com.exemple.traitement.writer; import com.exemple.traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import org.springframework.core.io.FileSystemResource; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { this.objectMapper = new ObjectMapper(); *// Initialisation de l'ObjectMapper pour la sérialisation JSON* } @Override public void write(List<? extends Person> items) throws Exception { try { *// Sérialisation de la liste d'objets Person en JSON et écriture dans le fichier* objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), items); } catch (IOException e) { throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } }

**2. Ajuster la configuration du Job et du Step**

Dans la configuration du job (BatchConfig.java), l'écrivain doit maintenant être configuré pour utiliser PersonWriter comme décrit précédemment. Nous n'avons pas besoin de changer cette partie puisque la logique de lecture et de traitement reste la même.

java

CopierModifier

package com.exemple.traitement.config; import com.exemple.traitement.model.Person; import com.exemple.traitement.processor.PersonProcessor; import com.exemple.traitement.reader.PersonReader; import com.exemple.traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.repository.JobRepository; import org.springframework.batch.core.repository.support.MapJobRepositoryFactoryBean; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemWriter; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.FixedLengthTokenizer; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; import org.springframework.batch.core.Job; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.core.io.FileSystemResource; import org.springframework.batch.item.file.transform.DefaultLineMapper; import org.springframework.batch.item.file.transform.LineTokenizer; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job job(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, ItemReader<Person> reader, ItemProcessor<Person, Person> processor, ItemWriter<Person> writer) { Step step = stepBuilderFactory.get("step1") .<Person, Person> chunk(10) .reader(reader) .processor(processor) .writer(writer) .build(); return jobBuilderFactory.get("job1") .incrementer(new RunIdIncrementer()) .start(step) .build(); } @Bean public ItemReader<Person> reader() { return new PersonReader(); } @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); } @Bean public ItemWriter<Person> writer() { return new PersonWriter(); } }

**3. Le fichier d'entrée (**people.txt**)**

Assurez-vous que le fichier d'entrée suit bien le format à partir duquel le PersonReader lira les données. Par exemple :

nginx

CopierModifier

John Doe 25 1234567890 Paris

Jane Smith 32 2345678901 Lyon

**4. Le fichier de sortie (**result.json**)**

Une fois que le batch sera exécuté, le fichier result.json contiendra les données sous forme de JSON. Le fichier ressemblera à ceci :

json

CopierModifier

[ { "nom": "John", "prenom": "Doe", "age": 25, "numCompte": "1234567890", "ville": "Paris" }, { "nom": "Jane", "prenom": "Smith", "age": 32, "numCompte": "2345678901", "ville": "Lyon" } ]

**5. Vérification des dépendances de**pom.xml

Assurez-vous que vous avez les dépendances nécessaires pour utiliser Jackson. Vous pouvez ajouter la dépendance suivante dans votre fichier pom.xml si elle n'est pas encore présente :

xml

CopierModifier

<dependencies> *<!-- Dépendance pour Spring Batch -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> *<!-- Assurez-vous d'utiliser la version appropriée -->* </dependency> *<!-- Dépendance pour Jackson -->* <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> <version>2.13.0</version> *<!-- Utilisez la version la plus récente -->* </dependency> *<!-- Dépendance pour Spring Boot (si nécessaire) -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> </dependencies>

**6. Test du Batch**

Lorsque vous lancez votre application, le batch lira les données du fichier people.txt, les traitera et écrira le résultat dans le fichier result.json au format JSON. Assurez-vous que la configuration de Spring Batch dans BatchConfig est correctement définie et que les chemins des fichiers sont corrects.

**Conclusion**

Avec ces modifications, vous serez en mesure de générer un fichier JSON comme sortie au lieu d'un fichier CSV. Le fichier result.json contiendra les données traitées sous forme de liste d'objets JSON. Si vous avez d'autres questions ou souhaitez des ajustements supplémentaires, n'hésitez pas à demander !

**Vous avez dit :**

je veux un code complet de tout le projet

**ChatGPT a dit :**

Bien sûr, voici un projet Spring Batch complet pour lire un fichier plat (format texte) contenant des informations de personnes, les traiter et générer un fichier de sortie au format JSON.

**Structure du projet**

css

CopierModifier

src

├── main

│ ├── java

│ │ └── com

│ │ └── exemple

│ │ └── traitement

│ │ ├── config

│ │ │ └── BatchConfig.java

│ │ ├── model

│ │ │ └── Person.java

│ │ ├── processor

│ │ │ └── PersonProcessor.java

│ │ ├── reader

│ │ │ └── PersonReader.java

│ │ ├── writer

│ │ │ └── PersonWriter.java

│ │ └── TraitementApplication.java

│ ├── resources

│ │ └── input

│ │ └── people.txt

│ │ └── output

│ │ └── result.json

│ ├── application.properties

├── pom.xml

**1. Fichier**pom.xml

Assurez-vous d'ajouter les dépendances nécessaires pour Spring Batch et Jackson.

xml

CopierModifier

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <groupId>com.exemple</groupId> <artifactId>traitement</artifactId> <version>1.0-SNAPSHOT</version> <packaging>jar</packaging> <dependencies> *<!-- Spring Boot -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> *<!-- Jackson for JSON -->* <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> <version>2.13.0</version> </dependency> *<!-- Spring Boot Starter for Logging -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-logging</artifactId> </dependency> *<!-- Spring Boot Starter for Configuration Processing -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter</artifactId> </dependency> *<!-- Spring Batch -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project>

**2. Fichier**application.properties

Voici un fichier application.properties de base pour Spring Boot.

properties

CopierModifier

spring.batch.job.enabled=true

**3. Classe**Person.java

Cette classe représente un objet Person.

java

CopierModifier

package com.exemple.traitement.model; public class Person { private String nom; private String prenom; private int age; private String numCompte; private String ville; *// Constructeurs, getters et setters* public Person(String nom, String prenom, int age, String numCompte, String ville) { this.nom = nom; this.prenom = prenom; this.age = age; this.numCompte = numCompte; this.ville = ville; } public String getNom() { return nom; } public void setNom(String nom) { this.nom = nom; } public String getPrenom() { return prenom; } public void setPrenom(String prenom) { this.prenom = prenom; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getNumCompte() { return numCompte; } public void setNumCompte(String numCompte) { this.numCompte = numCompte; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } }

**4. Classe**PersonReader.java

Le lecteur lira le fichier plat en utilisant une position fixe pour chaque champ.

java

CopierModifier

package com.exemple.traitement.reader; import com.exemple.traitement.model.Person; import org.springframework.batch.item.ItemReader; import org.springframework.core.io.FileSystemResource; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; public class PersonReader implements ItemReader<Person> { private static final String FILE\_PATH = "src/main/resources/input/people.txt"; private BufferedReader reader; public PersonReader() throws IOException { reader = new BufferedReader(new FileReader(FILE\_PATH)); } @Override public Person read() throws Exception { String line = reader.readLine(); if (line != null) { return mapLineToPerson(line); } return null; } private Person mapLineToPerson(String line) { String nom = line.substring(0, 15).trim(); String prenom = line.substring(16, 30).trim(); int age = Integer.parseInt(line.substring(31, 33).trim()); String numCompte = line.substring(34, 43).trim(); String ville = line.substring(44, 63).trim(); return new Person(nom, prenom, age, numCompte, ville); } }

**5. Classe**PersonProcessor.java

Cette classe sert à transformer ou à filtrer les objets Person s'il y a besoin de traitement. Dans ce cas, elle ne fait aucune transformation.

java

CopierModifier

package com.exemple.traitement.processor; import com.exemple.traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) throws Exception { return item; *// Aucun traitement, on passe simplement l'objet* } }

**6. Classe**PersonWriter.java

Cette classe va convertir les objets Person en JSON et les écrire dans un fichier.

java

CopierModifier

package com.exemple.traitement.writer; import com.exemple.traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { this.objectMapper = new ObjectMapper(); *// Utilisation de Jackson pour la sérialisation en JSON* } @Override public void write(List<? extends Person> items) throws Exception { try { *// Sérialiser la liste d'objets Person et les écrire dans un fichier JSON* objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), items); } catch (IOException e) { throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } }

**7. Classe**BatchConfig.java

La classe de configuration du batch.

java

CopierModifier

package com.exemple.traitement.config; import com.exemple.traitement.model.Person; import com.exemple.traitement.processor.PersonProcessor; import com.exemple.traitement.reader.PersonReader; import com.exemple.traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.repository.JobRepository; import org.springframework.batch.core.repository.support.MapJobRepositoryFactoryBean; import org.springframework.batch.core.Job; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job job(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, PersonReader reader, PersonProcessor processor, PersonWriter writer) { Step step = stepBuilderFactory.get("step1") .<Person, Person> chunk(10) .reader(reader) .processor(processor) .writer(writer) .build(); return jobBuilderFactory.get("job1") .incrementer(new RunIdIncrementer()) .start(step) .build(); } @Bean public PersonReader reader() throws Exception { return new PersonReader(); } @Bean public PersonProcessor processor() { return new PersonProcessor(); } @Bean public PersonWriter writer() { return new PersonWriter(); } }

**8. Classe**TraitementApplication.java

Le point d'entrée de l'application.

java

CopierModifier

package com.exemple.traitement; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class TraitementApplication { public static void main(String[] args) { SpringApplication.run(TraitementApplication.class, args); } }

**9. Fichier**people.txt

Voici un exemple de fichier plat d'entrée avec des données de personnes.

nginx

CopierModifier

John Doe 25 1234567890 Paris

Jane Smith 32 2345678901 Lyon

**10. Fichier de sortie**result.json

Après l'exécution du batch, le fichier result.json contiendra un format JSON similaire à ceci :

json

CopierModifier

[ { "nom": "John", "prenom": "Doe", "age": 25, "numCompte": "1234567890", "ville": "Paris" }, { "nom": "Jane", "prenom": "Smith", "age": 32, "numCompte": "2345678901", "ville": "Lyon" } ]

**Conclusion**

Avec ce projet complet, vous pouvez traiter un fichier plat contenant des informations sur des personnes, et à la fin du processus, vous obtenez un fichier result.json avec les données en format JSON. Vous pouvez exécuter cette application en utilisant mvn spring-boot:run ou en la lançant à partir d'un IDE comme IntelliJ IDEA ou Eclipse.

**Vous avez dit :**

C:\Users\n.sahil\Downloads\Traitement\Traitement\src\main\java\com\traitement\Traitement\config\BatchConfig.java:5:40 java: package com.traitement.Traitement.reader does not exist

**ChatGPT a dit :**

L'erreur que vous avez rencontrée indique que le package com.traitement.Traitement.reader est manquant ou mal défini dans le projet. Cela se produit généralement lorsque la déclaration du package dans un fichier ne correspond pas à la structure de votre projet.

Voyons comment corriger cela :

**1. Vérification des Packages**

L'erreur provient du fait que le package com.traitement.Traitement.reader n'est pas trouvé. Vous devez vérifier que tous les packages dans vos fichiers Java sont correctement définis.

**Assurez-vous que le package dans votre fichier**PersonReader.java**est correctement défini.**  
Si votre structure de projet est correcte, le package de PersonReader.java devrait correspondre à :

java

CopierModifier

package com.exemple.traitement.reader;

**2. Mise à jour des Classes**

Si la structure de votre projet est comme suit :

css

CopierModifier

src

└── main

└── java

└── com

└── exemple

└── traitement

├── reader

└── PersonReader.java

├── processor

└── PersonProcessor.java

├── writer

└── PersonWriter.java

├── model

└── Person.java

├── config

└── BatchConfig.java

└── TraitementApplication.java

Alors, vous devez vous assurer que la déclaration du package dans votre fichier BatchConfig.java est correcte pour correspondre à la structure réelle.

Dans BatchConfig.java, le bon import pour PersonReader serait :

java

CopierModifier

import com.exemple.traitement.reader.PersonReader;

**3. Vérification des autres classes**

* Dans PersonProcessor.java, assurez-vous que la déclaration du package est correcte :

java

CopierModifier

package com.exemple.traitement.processor;

* Dans PersonWriter.java, assurez-vous que le package est correct aussi :

java

CopierModifier

package com.exemple.traitement.writer;

**4. Modifications finales pour**BatchConfig.java

Voici la version correcte de BatchConfig.java avec les imports appropriés :

java

CopierModifier

package com.exemple.traitement.config; import com.exemple.traitement.model.Person; import com.exemple.traitement.processor.PersonProcessor; import com.exemple.traitement.reader.PersonReader; import com.exemple.traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.Job; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job job(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, PersonReader reader, PersonProcessor processor, PersonWriter writer) { Step step = stepBuilderFactory.get("step1") .<Person, Person> chunk(10) .reader(reader) .processor(processor) .writer(writer) .build(); return jobBuilderFactory.get("job1") .incrementer(new RunIdIncrementer()) .start(step) .build(); } @Bean public PersonReader reader() throws Exception { return new PersonReader(); } @Bean public PersonProcessor processor() { return new PersonProcessor(); } @Bean public PersonWriter writer() { return new PersonWriter(); } }

**5. Vérification du Fichier**PersonReader.java

Votre PersonReader.java devrait être situé dans le package com.exemple.traitement.reader et contenir le bon code comme suit :

java

CopierModifier

package com.exemple.traitement.reader; import com.exemple.traitement.model.Person; import org.springframework.batch.item.ItemReader; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; public class PersonReader implements ItemReader<Person> { private static final String FILE\_PATH = "src/main/resources/input/people.txt"; private BufferedReader reader; public PersonReader() throws IOException { reader = new BufferedReader(new FileReader(FILE\_PATH)); } @Override public Person read() throws Exception { String line = reader.readLine(); if (line != null) { return mapLineToPerson(line); } return null; } private Person mapLineToPerson(String line) { String nom = line.substring(0, 15).trim(); String prenom = line.substring(16, 30).trim(); int age = Integer.parseInt(line.substring(31, 33).trim()); String numCompte = line.substring(34, 43).trim(); String ville = line.substring(44, 63).trim(); return new Person(nom, prenom, age, numCompte, ville); } }

**6. Conclusion**

* Assurez-vous que tous les packages dans vos fichiers Java sont cohérents avec la structure du répertoire du projet.
* Modifiez les déclarations de package et les imports dans vos classes pour correspondre à la hiérarchie de votre projet.

Une fois que vous avez effectué ces changements, essayez de recompiler et de relancer le projet.

Si l'erreur persiste, assurez-vous également que votre environnement de développement est bien configuré et que le projet est bien construit (par exemple, en nettoyant et en reconstruisant le projet dans votre IDE ou avec Maven).

**Vous avez dit :**

C:\Users\n.sahil\Downloads\Traitement\Traitement\src\main\java\com\traitement\Traitement\config\BatchConfig.java:9:63 java: cannot find symbol symbol: class JobBuilderFactory location: package org.springframework.batch.core.configuration.annotation

**ChatGPT a dit :**

L'erreur que vous rencontrez, indiquant que JobBuilderFactory ne peut pas être trouvé, signifie que votre projet n'a pas correctement importé la dépendance Spring Batch nécessaire pour ce composant, ou que vous avez une erreur dans votre configuration des dépendances dans le fichier pom.xml.

**Étapes pour résoudre cette erreur :**

**1. Ajouter les dépendances Spring Batch**

Vous devez vous assurer que la dépendance Spring Batch est bien incluse dans votre pom.xml. Voici les dépendances essentielles pour Spring Batch :

xml

CopierModifier

<dependencies> *<!-- Spring Boot starter pour Spring Batch -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> *<!-- Spring Boot starter pour les Web Applications (si vous en avez besoin) -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency> *<!-- Pour travailler avec JSON, ajout de Jackson -->* <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> *<!-- Ajouter d'autres dépendances Spring Boot ou Spring Batch ici selon vos besoins -->* </dependencies>

Dans ce cas, la dépendance clé pour Spring Batch est spring-boot-starter-batch.

**2. Ajouter les imports corrects dans**BatchConfig.java

En fonction de la version de Spring Batch que vous utilisez, vous devez vous assurer que vous avez importé les bonnes classes. Voici un exemple de ce que vous devriez importer dans BatchConfig.java :

java

CopierModifier

package com.exemple.traitement.config; import com.exemple.traitement.model.Person; import com.exemple.traitement.processor.PersonProcessor; import com.exemple.traitement.reader.PersonReader; import com.exemple.traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.Job; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Job job(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory, PersonReader reader, PersonProcessor processor, PersonWriter writer) { Step step = stepBuilderFactory.get("step1") .<Person, Person> chunk(10) .reader(reader) .processor(processor) .writer(writer) .build(); return jobBuilderFactory.get("job1") .incrementer(new RunIdIncrementer()) .start(step) .build(); } @Bean public PersonReader reader() throws Exception { return new PersonReader(); } @Bean public PersonProcessor processor() { return new PersonProcessor(); } @Bean public PersonWriter writer() { return new PersonWriter(); } }

**3. Vérifier la version de Spring Batch**

Si vous utilisez une version plus ancienne de Spring Batch, certaines classes et méthodes peuvent avoir changé. Dans ce cas, assurez-vous d'utiliser la version compatible de Spring Batch. Voici un exemple de dépendance pour une version spécifique de Spring Batch (par exemple, 4.3.5):

xml

CopierModifier

<dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>4.3.5</version> </dependency>

Si vous utilisez Spring Boot avec spring-boot-starter-batch, il gère la version pour vous, donc il est préférable de laisser Spring Boot déterminer la version de Spring Batch.

**4. Rafraîchir ou reconstruire le projet**

Une fois ces changements effectués, essayez les étapes suivantes dans votre IDE :

1. **Nettoyez le projet**: (Dans IntelliJ IDEA, Build > Rebuild Project).
2. **Rafraîchissez les dépendances Maven**: Vous pouvez utiliser la commande suivante dans votre terminal pour Maven :

bash

CopierModifier

mvn clean install

Cela va réinstaller toutes les dépendances et s'assurer que les classes sont bien disponibles.

1. **Redémarrez l'IDE**: Si l'IDE ne détecte toujours pas les classes, essayez de redémarrer votre IDE.

**Conclusion**

1. **Assurez-vous que Spring Batch est correctement inclus dans votre**pom.xml.
2. **Vérifiez que vos imports dans**BatchConfig.java**sont corrects**.
3. **Rafraîchissez ou reconstruisez votre projet**.

Cela devrait résoudre l'erreur liée au JobBuilderFactory. Si l'erreur persiste, vous pouvez également vérifier que le bon SDK Java et la bonne version de Spring Boot sont utilisés dans votre projet.

**Vous avez dit :**

<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>3.4.4</version> <relativePath/> </parent> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>Traitement</name> <description>Demo project for Spring Boot with Batch processing</description> <properties> <java.version>17</java.version> </properties> <dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <!-- PostgreSQL Database Driver --> <dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency> <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <optional>true</optional> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </dependency> <dependency> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <configuration> <source>17</source> <target>17</target> <annotationProcessorPaths> <path> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </path> </annotationProcessorPaths> </configuration> </plugin> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project> je ne souhaite pas changer la version 3.4.4

**ChatGPT a dit :**

D'accord, je comprends que vous ne souhaitez pas changer la version de Spring Boot à 3.4.4. Le problème réside probablement dans la version de spring-batch-core qui doit être compatible avec la version de Spring Boot que vous utilisez.

Spring Boot 3.x est compatible avec Spring Batch 5.x, donc nous devons adapter les dépendances à cette version.

**Mise à jour des dépendances pour Spring Batch 5.x**

Dans ce cas, vous devez vous assurer que les dépendances de Spring Batch sont compatibles avec Spring Boot 3.4.4. Vous pouvez donc mettre à jour la dépendance spring-batch-core et d'autres dépendances liées à Spring Batch pour garantir la compatibilité avec Spring Boot 3.x.

Voici la version du fichier pom.xml corrigée :

xml

CopierModifier

<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>3.4.4</version> <relativePath/> </parent> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>Traitement</name> <description>Demo project for Spring Boot with Batch processing</description> <properties> <java.version>17</java.version> </properties> <dependencies> *<!-- Spring Boot Starter pour Spring Batch -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> *<!-- Spring Boot Starter Data JPA (si vous en avez besoin pour votre application) -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> *<!-- Driver PostgreSQL (si vous utilisez une base de données PostgreSQL) -->* <dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency> *<!-- Lombok pour éviter l'écriture des getters/setters -->* <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <optional>true</optional> </dependency> *<!-- Dépendance pour les tests -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> *<!-- Dépendance Spring Batch pour les tests -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-test</artifactId> <scope>test</scope> </dependency> *<!-- Jackson Databind pour manipuler le JSON -->* <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> *<!-- Spring Batch Core (5.x compatible avec Spring Boot 3.x) -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> *<!-- Utilisation de la version 5.x compatible avec Spring Boot 3.x -->* </dependency> *<!-- Plugin Maven pour Spring Boot -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </dependency> </dependencies> <build> <plugins> *<!-- Plugin pour la compilation avec Java 17 et Lombok -->* <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <configuration> <source>17</source> <target>17</target> <annotationProcessorPaths> <path> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </path> </annotationProcessorPaths> </configuration> </plugin> *<!-- Plugin Spring Boot pour Maven -->* <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project>

**Explication des changements :**

1. **Spring Batch Core version 5.x** : Comme Spring Boot 3.x utilise Spring Batch 5.x, j'ai ajouté la version 5.0.0 pour spring-batch-core et ses autres dépendances associées (comme spring-batch-test).
2. **Dépendances Spring Boot** : Vous conservez spring-boot-starter-batch, qui est la dépendance principale pour Spring Batch avec Spring Boot, et spring-boot-starter-data-jpa si vous avez besoin de la persistance avec JPA. Vous avez également spring-boot-starter-test pour vos tests.
3. **Lombok** : Pour éviter de créer des getters et setters manuellement dans vos classes, lombok est inclus comme dépendance optionnelle.
4. **PostgreSQL** : Si vous utilisez PostgreSQL, la dépendance postgresql est incluse avec le scope runtime.
5. **Jackson Databind** : Pour la gestion du format JSON (utilisé pour l'exportation des données au format JSON à la fin du traitement).
6. **Plugins Maven** : Les plugins nécessaires pour la compilation Java 17 avec Lombok et pour l'exécution de Spring Boot avec Maven.

**Étapes supplémentaires :**

1. **Mettre à jour votre IDE** : Une fois que vous avez mis à jour le pom.xml, assurez-vous de recharger les dépendances dans votre IDE (IntelliJ IDEA, Eclipse, etc.) pour que les classes Spring Batch soient correctement importées.
2. **Nettoyer et reconstruire le projet** : Pour vous assurer que toutes les dépendances sont prises en compte et que le projet se compile correctement, utilisez la commande Maven suivante :

bash

CopierModifier

mvn clean install

Cela devrait résoudre vos problèmes de compatibilité de version et faire en sorte que le projet fonctionne correctement avec Spring Boot 3.4.4 et Spring Batch 5.x. Si vous rencontrez d'autres erreurs, n'hésitez pas à me les signaler.

**Vous avez dit :**

je veux le pom sans commantaire

**ChatGPT a dit :**

Voici le fichier pom.xml sans les commentaires :

xml

CopierModifier

<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>3.4.4</version> <relativePath/> </parent> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>Traitement</name> <description>Demo project for Spring Boot with Batch processing</description> <properties> <java.version>17</java.version> </properties> <dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency> <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <optional>true</optional> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <configuration> <source>17</source> <target>17</target> <annotationProcessorPaths> <path> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </path> </annotationProcessorPaths> </configuration> </plugin> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project>

**Vous avez dit :**

Traitement .idea target Traitement .mvn wrapper src main java Traitement config BatchConfig model Person processor PersonProcessor reader PersonReader writer TraitementApplication resources input fichierV11.txt output application.properties test target .gitattributes .gitignore HELP.md mvnw mvnw.cmd pom.xml voici l'arborescence C:\Users\n.sahil\Downloads\Traitement\Traitement\src\main\java\com\traitement\Traitement\writer\PersonWriter.java:2:28 java: package com.traitement.model does not exist

**ChatGPT a dit :**

L'erreur que vous rencontrez signifie que le package com.traitement.model n'a pas été trouvé. Cela se produit généralement pour deux raisons possibles :

1. Le fichier Person.java n'existe pas dans le répertoire spécifié.
2. Il y a un problème de nommage ou de structure dans votre projet qui empêche la découverte du package et des classes.

**Vérification des étapes :**

1. **Vérifiez la présence du fichier**Person.java : Assurez-vous que le fichier Person.java est présent dans le répertoire suivant :

swift

CopierModifier

src/main/java/com/traitement/model/Person.java

1. **Vérifiez la déclaration du package dans**Person.java : Le fichier Person.java devrait commencer par une déclaration de package correcte, comme ceci :

java

CopierModifier

package com.traitement.model; public class Person { private String nom; private String prenom; private int age; private String nmCompte; private String ville; *// Getters and Setters* }

**Si le nom du package dans**Person.java**est incorrect** ou ne correspond pas au chemin du dossier dans lequel il se trouve, vous devez le corriger.

1. **Vérifiez l'importation de**Person**dans**PersonWriter.java : Assurez-vous que dans le fichier PersonWriter.java, vous avez correctement importé la classe Person :

java

CopierModifier

package com.traitement.writer; import com.traitement.model.Person; public class PersonWriter { *// Votre code* }

1. **Vérifiez la structure de votre projet** : La structure de répertoires doit respecter le nom du package Java. Par exemple, le fichier Person.java doit se trouver sous src/main/java/com/traitement/model/ et non sous un autre répertoire.
2. **Nettoyez et reconstruisez votre projet** : Après avoir effectué ces vérifications, essayez de nettoyer et de reconstruire votre projet à l'aide de Maven. Vous pouvez faire cela en exécutant les commandes suivantes dans votre terminal :
   * Pour nettoyer le projet :

bash

CopierModifier

mvn clean

* + Pour construire le projet :

bash

CopierModifier

mvn install

1. Cela va forcer Maven à recompiler et à vérifier toutes les dépendances et les classes.

**Résumé des étapes à suivre :**

1. Assurez-vous que le fichier Person.java est bien situé dans src/main/java/com/traitement/model/.
2. Vérifiez que la déclaration de package dans Person.java est correcte.
3. Vérifiez que vous avez importé la classe Person correctement dans PersonWriter.java.
4. Nettoyez et reconstruisez votre projet avec Maven (mvn clean install).

Cela devrait résoudre votre problème de package non trouvé. Si l'erreur persiste, assurez-vous qu'il n'y a pas de problème de structure de projet ou de chemin d'accès incorrect.

**Vous avez dit :**

package com.traitement.Traitement.writer; import com.traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { this.objectMapper = new ObjectMapper(); } @Override public void write(List<? extends Person> items) throws Exception { try { objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), items); } catch (IOException e) { throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } }

**ChatGPT a dit :**

Le code que vous avez fourni pour la classe PersonWriter semble correct dans l'ensemble. Il implémente l'interface ItemWriter<Person> de Spring Batch pour écrire une liste d'objets Person dans un fichier JSON. Toutefois, je vais vous donner quelques points à vérifier et améliorer pour être sûr que cela fonctionne correctement dans votre projet.

**Points à vérifier et améliorer :**

1. **Déclaration de package** : Assurez-vous que le package du fichier est bien aligné avec la structure des dossiers dans votre projet. Le fichier PersonWriter.java doit se trouver sous src/main/java/com/traitement/Traitement/writer/, et le package dans le fichier doit être déclaré comme suit :

java

CopierModifier

package com.traitement.Traitement.writer;

1. **Assurez-vous que**Person.java**est correctement défini** : Votre classe Person doit être dans le package com.traitement.model, et il doit être bien déclaré dans le fichier Person.java comme suit :

java

CopierModifier

package com.traitement.model; public class Person { private String nom; private String prenom; private int age; private String nmCompte; private String ville; *// Getters and Setters* }

1. **Utilisation de**ObjectMapper : L'utilisation de ObjectMapper pour convertir les objets Person en JSON semble correcte. Cependant, si vous souhaitez gérer des configurations supplémentaires ou personnaliser le format de sortie JSON (par exemple, gestion des dates, noms des champs, etc.), vous pouvez ajuster l'instance de ObjectMapper en conséquence. Mais pour un usage basique, la configuration actuelle est suffisante.
2. **Gestion de l'écriture dans le fichier** : Le fichier de sortie result.json est situé dans le dossier src/main/resources/output/. Vérifiez que ce répertoire existe bien. Si ce n'est pas le cas, vous pouvez le créer manuellement ou dans le code :
   * Dans le terminal, vous pouvez créer le dossier output :

bash

CopierModifier

mkdir -p src/main/resources/output

1. **Permissions d'écriture** : Assurez-vous que votre application a les droits nécessaires pour écrire dans le fichier result.json et que ce fichier n'est pas en lecture seule.

**Code complet de**PersonWriter.java**:**

Voici la version corrigée et complète de votre code avec quelques petits ajustements pour s'assurer que tout est bien configuré.

java

CopierModifier

package com.traitement.Traitement.writer; import com.traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { *// Initialisation de l'ObjectMapper pour la sérialisation en JSON* this.objectMapper = new ObjectMapper(); } @Override public void write(List<? extends Person> items) throws Exception { try { *// Sérialisation de la liste des objets Person en fichier JSON* objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), items); } catch (IOException e) { *// Gestion des erreurs d'écriture* throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } }

**Ce que vous devez vérifier dans votre projet :**

1. **Vérification de la structure de votre projet** :
   * Assurez-vous que le fichier PersonWriter.java est bien dans le répertoire suivant :

swift

CopierModifier

src/main/java/com/traitement/Traitement/writer/PersonWriter.java

1. **Vérification de l'importation de**Person**dans**PersonWriter.java : Si Person est dans le package com.traitement.model, vous devez l'importer comme suit :

java

CopierModifier

import com.traitement.model.Person;

1. **Assurez-vous que**Person.java**est bien dans**src/main/java/com/traitement/model/.
2. **Vérification de la configuration de Spring Batch** : Assurez-vous que votre BatchConfig.java et les autres configurations Spring Batch sont bien définies pour que ce writer soit utilisé dans votre job de traitement.

**Test :**

Une fois ces vérifications effectuées, vous pouvez tester votre job Spring Batch et vérifier si le fichier result.json est généré dans le répertoire src/main/resources/output/.

Si vous avez d'autres questions ou si vous avez besoin de plus d'aide sur la configuration de Spring Batch, n'hésitez pas à demander.

**Vous avez dit :**

package com.traitement.Traitement.model; public class Person { private String nom; private String prenoms; private int age; private String numcompte; private String ville; public Person(String nom, String prenoms, int age, String numcompte, String ville) { this.nom = nom; this.prenoms = prenoms; this.age = age; this.numcompte = numcompte; this.ville = ville; } public String getNom() { return nom; } public void setNom(String nom) { this.nom = nom; } public String getPrenoms() { return prenoms; } public void setPrenoms(String prenoms) { this.prenoms = prenoms; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getNumcompte() { return numcompte; } public void setNumcompte(String numcompte) { this.numcompte = numcompte; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } }

**ChatGPT a dit :**

Le code de la classe Person que vous avez fourni semble parfait pour représenter un individu avec les propriétés nécessaires pour votre projet. Il contient un constructeur pour initialiser ces propriétés, des getters et des setters pour chaque champ, ce qui est essentiel pour la sérialisation JSON dans votre PersonWriter.

Voici un résumé de ce que vous devez faire pour intégrer cette classe dans votre projet Spring Batch.

**Récapitulatif de la structure de la classe**Person**:**

La classe Person contient les champs suivants :

* **nom** : Le nom de la personne.
* **prenoms** : Les prénoms de la personne.
* **age** : L'âge de la personne.
* **numcompte** : Le numéro de compte de la personne.
* **ville** : La ville où habite la personne.

Les getters et setters permettent à Spring Batch d’accéder à ces propriétés et d'effectuer des transformations si nécessaire. De plus, la classe est déjà prête pour être sérialisée en JSON.

**Améliorations possibles**

Vous pourriez envisager d'utiliser des annotations de **lombok** pour simplifier la classe si vous souhaitez réduire le code boilerplate. Cela vous permettrait de remplacer les getters, setters et le constructeur par des annotations.

Exemple avec Lombok :

java

CopierModifier

package com.traitement.Traitement.model; import lombok.AllArgsConstructor; import lombok.Data; @Data @AllArgsConstructor public class Person { private String nom; private String prenoms; private int age; private String numcompte; private String ville; }

Cela éliminerait les méthodes getter et setter manuelles ainsi que le constructeur. Mais, bien sûr, cela nécessite d'ajouter Lombok comme dépendance dans votre pom.xml.

**Vérification de la structure du projet :**

Assurez-vous que la classe Person se trouve dans le répertoire suivant :

swift

CopierModifier

src/main/java/com/traitement/Traitement/model/Person.java

Et que le **package** de la classe correspond bien à l'arborescence du projet :

java

CopierModifier

package com.traitement.Traitement.model;

**Vérification du bon fonctionnement avec**PersonWriter**:**

Le fichier PersonWriter.java que vous avez écrit dans le répertoire src/main/java/com/traitement/Traitement/writer/ utilise la classe Person, donc assurez-vous que les imports dans PersonWriter.java sont corrects :

java

CopierModifier

import com.traitement.Traitement.model.Person;

**Test de sérialisation :**

Une fois que tout est bien en place, vous pouvez tester la sérialisation de vos objets Person en JSON avec Spring Batch pour vérifier si le fichier result.json est bien créé dans le répertoire output.

**Exemple de fichier JSON généré :**

Si vous avez un fichier people.txt avec des données telles que :

sql

CopierModifier

Doe John 30 1234567890 New York

Smith Jane 25 0987654321 Los Angeles

Le fichier result.json généré pourrait ressembler à ceci :

json

CopierModifier

[ { "nom": "Doe", "prenoms": "John", "age": 30, "numcompte": "1234567890", "ville": "New York" }, { "nom": "Smith", "prenoms": "Jane", "age": 25, "numcompte": "0987654321", "ville": "Los Angeles" } ]

**Conclusion :**

Votre classe Person est maintenant bien configurée pour être utilisée avec Spring Batch et PersonWriter. Assurez-vous que tous les fichiers de votre projet sont correctement organisés, que les chemins d'accès aux fichiers sont corrects, et que votre configuration Spring Batch est bien définie pour que l'écriture du fichier JSON fonctionne correctement.

Si vous avez d'autres questions ou souhaitez apporter des améliorations supplémentaires, n'hésitez pas à demander !

**Vous avez dit :**

import com.traitement.Traitement.model.Person;

**ChatGPT a dit :**

Il semble que vous essayez d'importer la classe Person dans l'un de vos fichiers Java, mais vous avez mentionné un problème potentiel avec l'importation.

Assurez-vous que l'importation est correctement formulée et que la classe Person est située au bon

**Vous avez dit :**

C:\Users\n.sahil\Downloads\Traitement\Traitement\src\main\java\com\traitement\Traitement\writer\PersonWriter.java:11:8 java: com.traitement.Traitement.writer.PersonWriter is not abstract and does not override abstract method write(org.springframework.batch.item.Chunk<? extends com.traitement.Traitement.model.Person>) in org.springframework.batch.item.ItemWriter

**ChatGPT a dit :**

L'erreur que vous obtenez indique que la méthode write dans la classe PersonWriter ne correspond pas à la signature attendue par l'interface ItemWriter. En effet, l'interface ItemWriter attend une méthode write qui prend un paramètre de type Chunk<? extends T>, où T est votre classe de modèle (Person dans ce cas).

**Solution**

Pour corriger cela, vous devez modifier la signature de la méthode write dans la classe PersonWriter pour qu'elle accepte un Chunk<? extends Person>, et non une List<? extends Person>. Un Chunk est une unité de travail dans Spring Batch, qui contient un sous-ensemble des éléments à traiter.

Voici la version corrigée de la classe PersonWriter :

**Code corrigé de**PersonWriter.java**:**

java

CopierModifier

package com.traitement.Traitement.writer; import com.traitement.Traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import org.springframework.batch.item.Chunk; import java.io.File; import java.io.IOException; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { this.objectMapper = new ObjectMapper(); } @Override public void write(Chunk<? extends Person> items) throws Exception { try { *// Convert Chunk to List* objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), items.getItems()); } catch (IOException e) { throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } }

**Explication des modifications :**

1. **Signature de la méthode**write :
   * Nous avons changé la méthode write pour accepter un Chunk<? extends Person> au lieu d'un List<? extends Person>.
   * La classe Chunk contient une collection d'éléments à traiter, et nous devons appeler items.getItems() pour obtenir la liste des éléments (dans ce cas, des instances de Person).
2. **Utilisation de**getItems() :
   * Nous récupérons les éléments du Chunk en appelant items.getItems() et ensuite nous utilisons Jackson (ObjectMapper) pour les écrire dans un fichier JSON.

**Pourquoi cette modification ?**

Spring Batch gère les éléments par morceaux ou **chunks** pour optimiser la gestion de la mémoire et le traitement par lots. Au lieu de traiter chaque élément un par un, Spring Batch gère des groupes d'éléments dans des Chunk et les envoie à l'écrivain pour un traitement en masse.

**Après cela :**

* Cette modification devrait résoudre l'erreur que vous avez rencontrée.
* Assurez-vous que vous avez bien configuré tous les autres composants de Spring Batch, comme le Job, le Step, le Reader, et le Processor.

Essayez cette version et dites-moi si tout fonctionne correctement !

**Vous avez dit :**

C:\Users\n.sahil\Downloads\jdk-17.0.12\_windows-x64\_bin\jdk-17.0.12\bin\java.exe -XX:TieredStopAtLevel=1 -Dspring.output.ansi.enabled=always -Dcom.sun.management.jmxremote -Dspring.jmx.enabled=true -Dspring.liveBeansView.mbeanDomain -Dspring.application.admin.enabled=true "-Dmanagement.endpoints.jmx.exposure.include=\*" "-javaagent:C:\Users\n.sahil\AppData\Local\JetBrains\IntelliJ IDEA 2025.1\lib\idea\_rt.jar=57222" -Dfile.encoding=UTF-8 -classpath C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-batch\3.4.4\spring-boot-starter-batch-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter\3.4.4\spring-boot-starter-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot\3.4.4\spring-boot-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-autoconfigure\3.4.4\spring-boot-autoconfigure-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-logging\3.4.4\spring-boot-starter-logging-3.4.4.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-classic\1.5.18\logback-classic-1.5.18.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-core\1.5.18\logback-core-1.5.18.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-to-slf4j\2.24.3\log4j-to-slf4j-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-api\2.24.3\log4j-api-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\jul-to-slf4j\2.0.17\jul-to-slf4j-2.0.17.jar;C:\Users\n.sahil\.m2\repository\jakarta\annotation\jakarta.annotation-api\2.1.1\jakarta.annotation-api-2.1.1.jar;C:\Users\n.sahil\.m2\repository\org\yaml\snakeyaml\2.3\snakeyaml-2.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-jdbc\3.4.4\spring-boot-starter-jdbc-3.4.4.jar;C:\Users\n.sahil\.m2\repository\com\zaxxer\HikariCP\5.1.0\HikariCP-5.1.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-data-jpa\3.4.4\spring-boot-starter-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\orm\hibernate-core\6.6.11.Final\hibernate-core-6.6.11.Final.jar;C:\Users\n.sahil\.m2\repository\jakarta\persistence\jakarta.persistence-api\3.1.0\jakarta.persistence-api-3.1.0.jar;C:\Users\n.sahil\.m2\repository\jakarta\transaction\jakarta.transaction-api\2.0.1\jakarta.transaction-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\jboss\logging\jboss-logging\3.6.1.Final\jboss-logging-3.6.1.Final.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\common\hibernate-commons-annotations\7.0.3.Final\hibernate-commons-annotations-7.0.3.Final.jar;C:\Users\n.sahil\.m2\repository\io\smallrye\jandex\3.2.0\jandex-3.2.0.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\classmate\1.7.0\classmate-1.7.0.jar;C:\Users\n.sahil\.m2\repository\net\bytebuddy\byte-buddy\1.15.11\byte-buddy-1.15.11.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-runtime\4.0.5\jaxb-runtime-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-core\4.0.5\jaxb-core-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\eclipse\angus\angus-activation\2.0.2\angus-activation-2.0.2.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\txw2\4.0.5\txw2-4.0.5.jar;C:\Users\n.sahil\.m2\repository\com\sun\istack\istack-commons-runtime\4.1.2\istack-commons-runtime-4.1.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\inject\jakarta.inject-api\2.0.1\jakarta.inject-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\antlr\antlr4-runtime\4.13.0\antlr4-runtime-4.13.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-jpa\3.4.4\spring-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-commons\3.4.4\spring-data-commons-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-orm\6.2.5\spring-orm-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\slf4j-api\2.0.17\slf4j-api-2.0.17.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aspects\6.2.5\spring-aspects-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\aspectj\aspectjweaver\1.9.23\aspectjweaver-1.9.23.jar;C:\Users\n.sahil\.m2\repository\org\postgresql\postgresql\42.7.5\postgresql-42.7.5.jar;C:\Users\n.sahil\.m2\repository\org\checkerframework\checker-qual\3.48.3\checker-qual-3.48.3.jar;C:\Users\n.sahil\.m2\repository\org\projectlombok\lombok\1.18.36\lombok-1.18.36.jar;C:\Users\n.sahil\.m2\repository\jakarta\xml\bind\jakarta.xml.bind-api\4.0.2\jakarta.xml.bind-api-4.0.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\activation\jakarta.activation-api\2.1.3\jakarta.activation-api-2.1.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-core\6.2.5\spring-core-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jcl\6.2.5\spring-jcl-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jdbc\6.2.5\spring-jdbc-6.2.5.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-databind\2.18.3\jackson-databind-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-annotations\2.18.3\jackson-annotations-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-core\2.18.3\jackson-core-2.18.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-core\5.0.0\spring-batch-core-5.0.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-infrastructure\5.2.2\spring-batch-infrastructure-5.2.2.jar;C:\Users\n.sahil\.m2\repository\org\springframework\retry\spring-retry\2.0.11\spring-retry-2.0.11.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aop\6.2.5\spring-aop-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-beans\6.2.5\spring-beans-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-context\6.2.5\spring-context-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-expression\6.2.5\spring-expression-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-tx\6.2.5\spring-tx-6.2.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-core\1.14.5\micrometer-core-1.14.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-commons\1.14.5\micrometer-commons-1.14.5.jar;C:\Users\n.sahil\.m2\repository\org\hdrhistogram\HdrHistogram\2.2.2\HdrHistogram-2.2.2.jar;C:\Users\n.sahil\.m2\repository\org\latencyutils\LatencyUtils\2.0.3\LatencyUtils-2.0.3.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-observation\1.14.5\micrometer-observation-1.14.5.jar com.traitement.Traitement.TraitementApplication . \_\_\_\_ \_ \_\_ \_ \_ /\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \ ( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_ | \ \ \ \ \\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) ) ' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / / =========|\_|==============|\_\_\_/=/\_/\_/\_/ :: Spring Boot :: (v3.4.4) 2025-04-23T14:14:54.725+01:00 INFO 17956 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : Starting TraitementApplication using Java 17.0.12 with PID 17956 (C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes started by n.sahil in C:\Users\n.sahil\Downloads\Traitement) 2025-04-23T14:14:54.729+01:00 INFO 17956 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : No active profile set, falling back to 1 default profile: "default" 2025-04-23T14:14:55.576+01:00 INFO 17956 --- [Traitement] [ main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode. 2025-04-23T14:14:55.613+01:00 INFO 17956 --- [Traitement] [ main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 20 ms. Found 0 JPA repository interfaces. 2025-04-23T14:14:56.074+01:00 WARN 17956 --- [Traitement] [ main] s.c.a.AnnotationConfigApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'entityManagerFactory' defined in class path resource [org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]: Failed to initialize dependency 'dataSourceScriptDatabaseInitializer' of LoadTimeWeaverAware bean 'entityManagerFactory': Error creating bean with name 'dataSourceScriptDatabaseInitializer' defined in class path resource [org/springframework/boot/autoconfigure/sql/init/DataSourceInitializationConfiguration.class]: Unsatisfied dependency expressed through method 'dataSourceScriptDatabaseInitializer' parameter 0: Error creating bean with name 'dataSource' defined in class path resource [org/springframework/boot/autoconfigure/jdbc/DataSourceConfiguration$Hikari.class]: Failed to instantiate [com.zaxxer.hikari.HikariDataSource]: Factory method 'dataSource' threw exception with message: Failed to determine a suitable driver class 2025-04-23T14:14:56.107+01:00 INFO 17956 --- [Traitement] [ main] .s.b.a.l.ConditionEvaluationReportLogger : Error starting ApplicationContext. To display the condition evaluation report re-run your application with 'debug' enabled. 2025-04-23T14:14:56.172+01:00 ERROR 17956 --- [Traitement] [ main] o.s.b.d.LoggingFailureAnalysisReporter : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* APPLICATION FAILED TO START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Description: Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured. Reason: Failed to determine a suitable driver class Action: Consider the following: If you want an embedded database (H2, HSQL or Derby), please put it on the classpath. If you have database settings to be loaded from a particular profile you may need to activate it (no profiles are currently active). Process finished with exit code 1

**ChatGPT a dit :**

L'erreur que vous rencontrez est liée à la configuration de votre **DataSource** dans votre application Spring Boot. Le message d'erreur indique qu'il n'a pas pu déterminer une classe de pilote appropriée pour votre base de données, car l'URL de la base de données n'a pas été spécifiée.

**Erreur :**

vbnet

CopierModifier

Failed to configure a DataSource: *'url' attribute is not specified and no embedded datasource could be configured.*

Reason: Failed to determine a suitable driver class.

**Causes possibles :**

1. **Configuration manquante de l'URL de la base de données** dans le fichier application.properties ou application.yml.
2. **Absence du pilote JDBC de la base de données** sur le classpath, bien que vous ayez déjà inclus PostgreSQL dans les dépendances Maven.

**Solutions possibles :**

**1. Vérifier la configuration dans**application.properties**ou**application.yml

Si vous utilisez PostgreSQL comme base de données, vous devez configurer les informations de connexion à la base de données dans le fichier application.properties situé dans le dossier src/main/resources.

Ajoutez ou modifiez les lignes suivantes dans application.properties :

properties

CopierModifier

spring.datasource.url=jdbc:postgresql://localhost:5432/nom\_de\_votre\_base\_de\_donnees spring.datasource.username=nom\_utilisateur spring.datasource.password=mot\_de\_passe spring.datasource.driver-class-name=org.postgresql.Driver spring.datasource.jpa.hibernate.ddl-auto=update spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect

* Remplacez nom\_de\_votre\_base\_de\_donnees par le nom de votre base de données PostgreSQL.
* Remplacez nom\_utilisateur et mot\_de\_passe par vos informations d'identification.

Si vous utilisez un autre SGBD (comme MySQL, H2, etc.), vous devrez adapter l'URL et le pilote en conséquence.

**2. Assurez-vous que la dépendance PostgreSQL est incluse**

Dans votre fichier pom.xml, vous avez déjà la dépendance PostgreSQL, donc normalement, elle est correctement incluse. Vérifiez que la version est compatible avec votre base de données et la configuration.

Voici la dépendance PostgreSQL que vous avez dans le pom.xml :

xml

CopierModifier

<dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency>

Cela semble correct. Cependant, si vous utilisez un autre type de base de données, n'oubliez pas de changer la dépendance en conséquence.

**3. Vérification du**HikariCP

Spring Boot utilise **HikariCP** comme pool de connexions par défaut, qui est compatible avec PostgreSQL. Si l'URL ou les informations d'identification sont manquantes, il échouera à créer une connexion.

**4. Activer le profil de développement (optionnel)**

Si vous avez plusieurs configurations de base de données en fonction de différents profils (par exemple, développement, production), vous pouvez activer le profil correspondant en ajoutant cette ligne dans application.properties :

properties

CopierModifier

spring.profiles.active=dev

Ou, si vous utilisez application.yml :

yaml

CopierModifier

spring: profiles: active: dev

**Exemple complet de**application.properties**:**

properties

CopierModifier

# Config PostgreSQL spring.datasource.url=jdbc:postgresql://localhost:5432/ma\_base\_de\_donnees spring.datasource.username=postgres spring.datasource.password=mon\_mot\_de\_passe spring.datasource.driver-class-name=org.postgresql.Driver spring.datasource.jpa.hibernate.ddl-auto=update spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect # Configuration Spring Batch spring.batch.initialize-schema=always

**Conclusion :**

1. Vérifiez que l'URL de la base de données et les informations d'identification sont correctes dans application.properties ou application.yml.
2. Si vous utilisez PostgreSQL, assurez-vous que le pilote est correctement configuré (ce que semble être le cas dans votre pom.xml).
3. Essayez de redémarrer l'application après avoir fait ces modifications.

Si vous avez d'autres questions ou si l'erreur persiste, n'hésitez pas à partager plus de détails !

**Vous avez dit :**

mais moi je ne veux pas utiliser une base de donner

**ChatGPT a dit :**

Ah, d'accord ! Si vous ne souhaitez pas utiliser de base de données et que l'erreur provient de la configuration de Spring Boot qui cherche une source de données (DataSource), vous pouvez désactiver la configuration automatique de la source de données.

Spring Boot tente de configurer une DataSource lorsqu'il détecte une configuration liée à la base de données dans votre projet (par exemple, si vous avez les dépendances de Spring Data JPA ou un fichier application.properties qui inclut des configurations de base de données).

**Solution : Désactiver la configuration automatique de la base de données**

Vous pouvez désactiver la configuration automatique de la base de données en ajoutant la propriété suivante dans le fichier application.properties ou application.yml :

**Dans**application.properties**:**

properties

CopierModifier

spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration

**Ou dans**application.yml**:**

yaml

CopierModifier

spring: autoconfigure: exclude: org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration

**Explication :**

* spring.autoconfigure.exclude : Cette propriété permet d'exclure des configurations automatiques spécifiques. En l'occurrence, org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration désactive la configuration de la source de données qui essaie de se connecter à une base de données.

Cela permettra à votre application de ne pas essayer de se connecter à une base de données, et donc d'éviter l'erreur liée à la base de données.

**Vérifiez vos dépendances :**

Assurez-vous que vous n'avez pas de dépendances inutiles dans votre pom.xml qui nécessitent une base de données, telles que celles liées à **Spring Data JPA** ou à des pilotes de bases de données.

Si vous n'utilisez pas de base de données, vous pouvez supprimer la dépendance suivante :

xml

CopierModifier

<dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency>

**Exemple de**pom.xml**simplifié (sans base de données) :**

Voici à quoi pourrait ressembler un fichier pom.xml simplifié sans dépendances liées à la base de données :

xml

CopierModifier

<dependencies> *<!-- Dépendance Spring Boot Starter Batch pour le traitement par lots -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> *<!-- Dépendance pour la lecture/écriture de fichiers JSON avec Jackson -->* <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> *<!-- Dépendance Spring Boot Starter Web (si vous avez besoin de fonctionnalités web, par exemple un API REST) -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency> *<!-- Autres dépendances nécessaires pour votre projet... -->* </dependencies>

**Conclusion :**

En désactivant la configuration automatique de la base de données (DataSourceAutoConfiguration) et en retirant les dépendances inutiles liées à la base de données, vous évitez l'erreur et pouvez continuer avec votre traitement par lots sans base de données.

**Vous avez dit :**

ecoute bien je veux ce projet mais sans base de donnée change ce qu'il faut pour obtenir à la fin un fichier json

**ChatGPT a dit :**

D'accord, je comprends mieux maintenant. Vous voulez un projet Spring Batch qui traite des données et génère un fichier JSON à la fin sans utiliser de base de données.

Voici comment vous pouvez configurer votre projet Spring Batch sans base de données et générer un fichier JSON en sortie.

**1. Dépendances du**pom.xml

Nous allons garder uniquement les dépendances nécessaires pour Spring Batch, Jackson pour la gestion du JSON, et les autres dépendances utiles pour votre projet. Voici un pom.xml sans les dépendances liées à une base de données.

xml

CopierModifier

<dependencies> *<!-- Dépendance Spring Boot Starter Batch pour le traitement par lots -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> *<!-- Dépendance Spring Boot Starter Web pour générer un fichier JSON en sortie -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency> *<!-- Dépendance pour la gestion des fichiers JSON avec Jackson -->* <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> *<!-- Dépendance pour SLF4J (loggers) -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-logging</artifactId> </dependency> *<!-- Dépendance pour la gestion des fichiers d'entrée/sortie (si vous utilisez des fichiers textes comme entrée) -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> </dependency> </dependencies>

**2. Configuration de Spring Batch**

Maintenant, nous allons configurer Spring Batch pour qu'il traite des données sans nécessiter de base de données. Vous aurez besoin de la configuration suivante pour les jobs Spring Batch :

BatchConfig.java**: Configuration de Spring Batch**

Nous allons configurer un job Spring Batch qui lit des données depuis un fichier texte, les traite et les écrit dans un fichier JSON.

java

CopierModifier

package com.traitement.Traitement.config; import com.traitement.Traitement.model.Person; import com.traitement.Traitement.processor.PersonProcessor; import com.traitement.Traitement.reader.PersonReader; import com.traitement.Traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.BatchConfigurer; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.core.io.FileSystemResource; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemWriter; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public ItemReader<Person> reader() { return new PersonReader(); *// Classe qui lit depuis un fichier* } @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); *// Classe qui traite les données* } @Bean public ItemWriter<Person> writer() { return new PersonWriter(); *// Classe qui écrit dans un fichier JSON* } @Bean public Step step1(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) .reader(reader()) .processor(processor()) .writer(writer()) .build(); } @Bean public Job processJob(JobBuilderFactory jobBuilderFactory, Step step1) { return jobBuilderFactory.get("processJob") .start(step1) .build(); } }

**3. Création du Reader (PersonReader.java)**

Le ItemReader lira les données depuis un fichier texte. Il pourrait ressembler à ceci :

java

CopierModifier

package com.traitement.Traitement.reader; import com.traitement.Traitement.model.Person; import org.springframework.batch.item.ItemReader; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; public class PersonReader implements ItemReader<Person> { private final String inputFile = "src/main/resources/input/fichierV11.txt"; private BufferedReader reader; private String currentLine; public PersonReader() { try { reader = new BufferedReader(new FileReader(inputFile)); } catch (IOException e) { e.printStackTrace(); } } @Override public Person read() throws Exception { currentLine = reader.readLine(); if (currentLine == null) { return null; *// Fin du fichier* } String[] values = currentLine.split(","); return new Person(values[0], values[1], Integer.parseInt(values[2]), values[3], values[4]); } public void close() throws IOException { if (reader != null) { reader.close(); } } }

**4. Création du Processor (PersonProcessor.java)**

Le ItemProcessor va transformer les données si nécessaire, ou les laisser telles quelles.

java

CopierModifier

package com.traitement.Traitement.processor; import com.traitement.Traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) throws Exception { *// Si vous avez des transformations à appliquer aux données, vous pouvez le faire ici* return item; } }

**5. Création du Writer (PersonWriter.java)**

Le ItemWriter écrit les données traitées dans un fichier JSON.

java

CopierModifier

package com.traitement.Traitement.writer; import com.traitement.Traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { this.objectMapper = new ObjectMapper(); } @Override public void write(List<? extends Person> items) throws Exception { try { objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), items); } catch (IOException e) { throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } }

**6. Application principale (TraitementApplication.java)**

Votre classe principale qui lance l'application Spring Batch.

java

CopierModifier

package com.traitement.Traitement; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class TraitementApplication { public static void main(String[] args) { SpringApplication.run(TraitementApplication.class, args); } }

**7. Fichier de configuration (**application.properties**)**

Enfin, vous devez vous assurer que Spring Boot ne tente pas de configurer une base de données. Ajoutez cette ligne dans votre fichier application.properties pour exclure la configuration automatique de la base de données :

properties

CopierModifier

spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration

**8. Structure des fichiers d'entrée et de sortie**

* Votre fichier d'entrée fichierV11.txt doit être dans le répertoire src/main/resources/input/.
* Le fichier de sortie result.json sera généré dans le répertoire src/main/resources/output/.

**Conclusion**

Avec cette configuration, vous avez un projet Spring Batch qui lit un fichier texte fichierV11.txt, traite les données et génère un fichier JSON result.json en sortie, **sans utiliser de base de données**. Vous pouvez étendre cette structure selon vos besoins, comme ajouter des transformations supplémentaires ou personnaliser la lecture/écriture des données.

**Vous avez dit :**

je veux un code sans commentaire

**ChatGPT a dit :**

D'accord, voici le code sans les commentaires.

**1.**pom.xml

xml

CopierModifier

<dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency> <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-logging</artifactId> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> </dependency> </dependencies>

**2.**BatchConfig.java

java

CopierModifier

package com.traitement.Traitement.config; import com.traitement.Traitement.model.Person; import com.traitement.Traitement.processor.PersonProcessor; import com.traitement.Traitement.reader.PersonReader; import com.traitement.Traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.BatchConfigurer; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.core.io.FileSystemResource; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemWriter; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public ItemReader<Person> reader() { return new PersonReader(); } @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); } @Bean public ItemWriter<Person> writer() { return new PersonWriter(); } @Bean public Step step1(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) .reader(reader()) .processor(processor()) .writer(writer()) .build(); } @Bean public Job processJob(JobBuilderFactory jobBuilderFactory, Step step1) { return jobBuilderFactory.get("processJob") .start(step1) .build(); } }

**3.**PersonReader.java

java

CopierModifier

package com.traitement.Traitement.reader; import com.traitement.Traitement.model.Person; import org.springframework.batch.item.ItemReader; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; public class PersonReader implements ItemReader<Person> { private final String inputFile = "src/main/resources/input/fichierV11.txt"; private BufferedReader reader; private String currentLine; public PersonReader() { try { reader = new BufferedReader(new FileReader(inputFile)); } catch (IOException e) { e.printStackTrace(); } } @Override public Person read() throws Exception { currentLine = reader.readLine(); if (currentLine == null) { return null; } String[] values = currentLine.split(","); return new Person(values[0], values[1], Integer.parseInt(values[2]), values[3], values[4]); } public void close() throws IOException { if (reader != null) { reader.close(); } } }

**4.**PersonProcessor.java

java

CopierModifier

package com.traitement.Traitement.processor; import com.traitement.Traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) throws Exception { return item; } }

**5.**PersonWriter.java

java

CopierModifier

package com.traitement.Traitement.writer; import com.traitement.Traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { this.objectMapper = new ObjectMapper(); } @Override public void write(List<? extends Person> items) throws Exception { try { objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), items); } catch (IOException e) { throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } }

**6.**TraitementApplication.java

java

CopierModifier

package com.traitement.Traitement; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class TraitementApplication { public static void main(String[] args) { SpringApplication.run(TraitementApplication.class, args); } }

**7.**application.properties

properties

CopierModifier

spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration

**Structure des fichiers :**

* Fichier d'entrée fichierV11.txt dans src/main/resources/input/.
* Fichier de sortie result.json sera généré dans src/main/resources/output/.

Avec ce code, vous avez un projet Spring Batch qui lira un fichier texte, traitera les données et écrira le résultat dans un fichier JSON, sans utiliser de base de données.

**Vous avez dit :**

C:\Users\n.sahil\Downloads\Traitement\Traitement\src\main\java\com\traitement\Traitement\writer\PersonWriter.java:11:8 java: com.traitement.Traitement.writer.PersonWriter is not abstract and does not override abstract method write(org.springframework.batch.item.Chunk<? extends com.traitement.Traitement.model.Person>) in org.springframework.batch.item.ItemWriter

**ChatGPT a dit :**

L'erreur que vous rencontrez indique que la classe PersonWriter doit implémenter la méthode write de l'interface ItemWriter, mais actuellement, vous n'avez pas correctement implémenté cette méthode selon la signature attendue. L'interface ItemWriter attend une méthode qui prend un objet Chunk<? extends T> en paramètre.

Voici la correction à apporter :

**1. Corriger**PersonWriter**pour implémenter correctement**ItemWriter

L'interface ItemWriter demande une méthode write(Chunk<? extends T> items), et non write(List<? extends T> items). Par conséquent, vous devez modifier votre méthode write pour accepter un objet Chunk<Person> et ensuite écrire ces objets dans un fichier JSON.

**Nouvelle version de**PersonWriter.java

java

CopierModifier

package com.traitement.Traitement.writer; import com.traitement.Traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import org.springframework.batch.item.Chunk; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { this.objectMapper = new ObjectMapper(); } @Override public void write(Chunk<? extends Person> items) throws Exception { try { List<? extends Person> personList = items.getItems(); objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), personList); } catch (IOException e) { throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } }

**Explications des changements :**

1. **Signature de la méthode**write :
   * Au lieu de prendre List<? extends Person> items, la méthode write prend maintenant un paramètre de type Chunk<? extends Person>.
   * Chunk est une partie d'un lot de données traitées, et la méthode getItems() permet d'obtenir la liste des éléments de ce chunk.
2. **Traitement du**Chunk :
   * Nous utilisons la méthode getItems() de Chunk pour récupérer la liste des objets Person dans le chunk, puis nous écrivons cette liste dans un fichier JSON.

**2. Vérification du fichier JSON généré**

Une fois que cette modification est effectuée, votre PersonWriter écrira les objets dans un fichier JSON à chaque exécution du batch.

Cela devrait résoudre l'erreur et permettre au traitement de se faire correctement.

**Vous avez dit :**

C:\Users\n.sahil\Downloads\jdk-17.0.12\_windows-x64\_bin\jdk-17.0.12\bin\java.exe -XX:TieredStopAtLevel=1 -Dspring.output.ansi.enabled=always -Dcom.sun.management.jmxremote -Dspring.jmx.enabled=true -Dspring.liveBeansView.mbeanDomain -Dspring.application.admin.enabled=true "-Dmanagement.endpoints.jmx.exposure.include=\*" "-javaagent:C:\Users\n.sahil\AppData\Local\JetBrains\IntelliJ IDEA 2025.1\lib\idea\_rt.jar=57361" -Dfile.encoding=UTF-8 -classpath C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-batch\3.4.4\spring-boot-starter-batch-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter\3.4.4\spring-boot-starter-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot\3.4.4\spring-boot-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-autoconfigure\3.4.4\spring-boot-autoconfigure-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-logging\3.4.4\spring-boot-starter-logging-3.4.4.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-classic\1.5.18\logback-classic-1.5.18.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-core\1.5.18\logback-core-1.5.18.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-to-slf4j\2.24.3\log4j-to-slf4j-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-api\2.24.3\log4j-api-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\jul-to-slf4j\2.0.17\jul-to-slf4j-2.0.17.jar;C:\Users\n.sahil\.m2\repository\jakarta\annotation\jakarta.annotation-api\2.1.1\jakarta.annotation-api-2.1.1.jar;C:\Users\n.sahil\.m2\repository\org\yaml\snakeyaml\2.3\snakeyaml-2.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-jdbc\3.4.4\spring-boot-starter-jdbc-3.4.4.jar;C:\Users\n.sahil\.m2\repository\com\zaxxer\HikariCP\5.1.0\HikariCP-5.1.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-data-jpa\3.4.4\spring-boot-starter-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\orm\hibernate-core\6.6.11.Final\hibernate-core-6.6.11.Final.jar;C:\Users\n.sahil\.m2\repository\jakarta\persistence\jakarta.persistence-api\3.1.0\jakarta.persistence-api-3.1.0.jar;C:\Users\n.sahil\.m2\repository\jakarta\transaction\jakarta.transaction-api\2.0.1\jakarta.transaction-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\jboss\logging\jboss-logging\3.6.1.Final\jboss-logging-3.6.1.Final.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\common\hibernate-commons-annotations\7.0.3.Final\hibernate-commons-annotations-7.0.3.Final.jar;C:\Users\n.sahil\.m2\repository\io\smallrye\jandex\3.2.0\jandex-3.2.0.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\classmate\1.7.0\classmate-1.7.0.jar;C:\Users\n.sahil\.m2\repository\net\bytebuddy\byte-buddy\1.15.11\byte-buddy-1.15.11.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-runtime\4.0.5\jaxb-runtime-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-core\4.0.5\jaxb-core-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\eclipse\angus\angus-activation\2.0.2\angus-activation-2.0.2.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\txw2\4.0.5\txw2-4.0.5.jar;C:\Users\n.sahil\.m2\repository\com\sun\istack\istack-commons-runtime\4.1.2\istack-commons-runtime-4.1.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\inject\jakarta.inject-api\2.0.1\jakarta.inject-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\antlr\antlr4-runtime\4.13.0\antlr4-runtime-4.13.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-jpa\3.4.4\spring-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-commons\3.4.4\spring-data-commons-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-orm\6.2.5\spring-orm-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\slf4j-api\2.0.17\slf4j-api-2.0.17.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aspects\6.2.5\spring-aspects-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\aspectj\aspectjweaver\1.9.23\aspectjweaver-1.9.23.jar;C:\Users\n.sahil\.m2\repository\org\postgresql\postgresql\42.7.5\postgresql-42.7.5.jar;C:\Users\n.sahil\.m2\repository\org\checkerframework\checker-qual\3.48.3\checker-qual-3.48.3.jar;C:\Users\n.sahil\.m2\repository\org\projectlombok\lombok\1.18.36\lombok-1.18.36.jar;C:\Users\n.sahil\.m2\repository\jakarta\xml\bind\jakarta.xml.bind-api\4.0.2\jakarta.xml.bind-api-4.0.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\activation\jakarta.activation-api\2.1.3\jakarta.activation-api-2.1.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-core\6.2.5\spring-core-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jcl\6.2.5\spring-jcl-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jdbc\6.2.5\spring-jdbc-6.2.5.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-databind\2.18.3\jackson-databind-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-annotations\2.18.3\jackson-annotations-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-core\2.18.3\jackson-core-2.18.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-core\5.0.0\spring-batch-core-5.0.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-infrastructure\5.2.2\spring-batch-infrastructure-5.2.2.jar;C:\Users\n.sahil\.m2\repository\org\springframework\retry\spring-retry\2.0.11\spring-retry-2.0.11.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aop\6.2.5\spring-aop-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-beans\6.2.5\spring-beans-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-context\6.2.5\spring-context-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-expression\6.2.5\spring-expression-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-tx\6.2.5\spring-tx-6.2.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-core\1.14.5\micrometer-core-1.14.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-commons\1.14.5\micrometer-commons-1.14.5.jar;C:\Users\n.sahil\.m2\repository\org\hdrhistogram\HdrHistogram\2.2.2\HdrHistogram-2.2.2.jar;C:\Users\n.sahil\.m2\repository\org\latencyutils\LatencyUtils\2.0.3\LatencyUtils-2.0.3.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-observation\1.14.5\micrometer-observation-1.14.5.jar com.traitement.Traitement.TraitementApplication . \_\_\_\_ \_ \_\_ \_ \_ /\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \ ( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_ | \ \ \ \ \\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) ) ' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / / =========|\_|==============|\_\_\_/=/\_/\_/\_/ :: Spring Boot :: (v3.4.4) 2025-04-23T14:21:58.067+01:00 INFO 18832 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : Starting TraitementApplication using Java 17.0.12 with PID 18832 (C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes started by n.sahil in C:\Users\n.sahil\Downloads\Traitement) 2025-04-23T14:21:58.072+01:00 INFO 18832 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : No active profile set, falling back to 1 default profile: "default" 2025-04-23T14:21:58.600+01:00 INFO 18832 --- [Traitement] [ main] o.s.b.c.c.annotation.BatchRegistrar : Finished Spring Batch infrastructure beans configuration in 8 ms. java.io.FileNotFoundException: src\main\resources\input\fichierV11.txt (Le chemin d’accès spécifié est introuvable) at java.base/java.io.FileInputStream.open0(Native Method) at java.base/java.io.FileInputStream.open(FileInputStream.java:216) at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157) at java.base/java.io.FileInputStream.<init>(FileInputStream.java:111) at java.base/java.io.FileReader.<init>(FileReader.java:60) at com.traitement.Traitement.reader.PersonReader.<init>(PersonReader.java:18) at com.traitement.Traitement.config.BatchConfig.reader(BatchConfig.java:24) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$0.CGLIB$reader$0(<generated>) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$FastClass$$1.invoke(<generated>) at org.springframework.cglib.proxy.MethodProxy.invokeSuper(MethodProxy.java:258) at org.springframework.context.annotation.ConfigurationClassEnhancer$BeanMethodInterceptor.intercept(ConfigurationClassEnhancer.java:370) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$0.reader(<generated>) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.base/java.lang.reflect.Method.invoke(Method.java:568) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.lambda$instantiate$0(SimpleInstantiationStrategy.java:171) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiateWithFactoryMethod(SimpleInstantiationStrategy.java:88) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiate(SimpleInstantiationStrategy.java:168) at org.springframework.beans.factory.support.ConstructorResolver.instantiate(ConstructorResolver.java:653) at org.springframework.beans.factory.support.ConstructorResolver.instantiateUsingFactoryMethod(ConstructorResolver.java:489) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.instantiateUsingFactoryMethod(AbstractAutowireCapableBeanFactory.java:1361) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBeanInstance(AbstractAutowireCapableBeanFactory.java:1191) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:563) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:523) at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:339) at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:347) at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:337) at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:202) at org.springframework.beans.factory.support.DefaultListableBeanFactory.instantiateSingleton(DefaultListableBeanFactory.java:1155) at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingleton(DefaultListableBeanFactory.java:1121) at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:1056) at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:987) at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:627) at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:752) at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:439) at org.springframework.boot.SpringApplication.run(SpringApplication.java:318) at org.springframework.boot.SpringApplication.run(SpringApplication.java:1361) at org.springframework.boot.SpringApplication.run(SpringApplication.java:1350) at com.traitement.Traitement.TraitementApplication.main(TraitementApplication.java:11) 2025-04-23T14:21:59.447+01:00 WARN 18832 --- [Traitement] [ main] s.c.a.AnnotationConfigApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'step1' defined in class path resource [com/traitement/Traitement/config/BatchConfig.class]: Unsatisfied dependency expressed through method 'step1' parameter 0: No qualifying bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' available: expected at least 1 bean which qualifies as autowire candidate. Dependency annotations: {} 2025-04-23T14:21:59.464+01:00 INFO 18832 --- [Traitement] [ main] .s.b.a.l.ConditionEvaluationReportLogger : Error starting ApplicationContext. To display the condition evaluation report re-run your application with 'debug' enabled. 2025-04-23T14:21:59.513+01:00 ERROR 18832 --- [Traitement] [ main] o.s.b.d.LoggingFailureAnalysisReporter : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* APPLICATION FAILED TO START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Description: Parameter 0 of method step1 in com.traitement.Traitement.config.BatchConfig required a bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' that could not be found. Action: Consider defining a bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' in your configuration. Process finished with exit code 1

**ChatGPT a dit :**

L'erreur que vous obtenez indique que le StepBuilderFactory requis pour la configuration du batch n'a pas été trouvé. Cela se produit généralement lorsque Spring Batch n'est pas correctement configuré dans votre application.

**Voici quelques étapes à suivre pour résoudre ce problème :**

1. **Vérifier les dépendances Spring Batch** : Assurez-vous que toutes les dépendances nécessaires à Spring Batch sont bien présentes dans votre fichier pom.xml si vous utilisez Maven.

Voici un exemple de dépendances Spring Batch dans votre pom.xml :

xml

CopierModifier

<dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> *<!-- Ajustez la version si nécessaire -->* </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> <version>3.4.4</version> *<!-- Ajustez la version si nécessaire -->* </dependency>

1. **Configurer correctement Spring Batch** : Pour que Spring Batch fonctionne, vous devez déclarer certains beans, notamment ceux qui sont responsables de la création des étapes (Step), des jobs (Job), et de la gestion de la transaction. Voici un exemple de configuration de Spring Batch dans la classe BatchConfig.

**Exemple de configuration pour Spring Batch (avec**StepBuilderFactory**et**JobBuilderFactory**):**

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.core.configuration.annotation.BatchConfiguration; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { private final JobBuilderFactory jobBuilderFactory; private final StepBuilderFactory stepBuilderFactory; public BatchConfig(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory) { this.jobBuilderFactory = jobBuilderFactory; this.stepBuilderFactory = stepBuilderFactory; } @Bean public Job step1Job() { return jobBuilderFactory.get("step1Job") .start(step1()) .build(); } @Bean public Step step1() { return stepBuilderFactory.get("step1") .<InputType, OutputType>chunk(10) .reader(reader()) *// Assurez-vous que le reader est bien configuré* .processor(processor()) *// Assurez-vous que le processor est bien configuré* .writer(writer()) *// Assurez-vous que le writer est bien configuré* .build(); } *// Configurez votre reader, processor et writer* *// Assurez-vous que ces beans sont bien déclarés* public ItemReader<InputType> reader() { return new PersonReader(); } public ItemProcessor<InputType, OutputType> processor() { return new PersonProcessor(); } public ItemWriter<OutputType> writer() { return new PersonWriter(); } }

**Explications :**

* @EnableBatchProcessing : Cette annotation active le support Spring Batch dans votre application et configure les beans nécessaires pour le traitement par lots.
* JobBuilderFactory**et**StepBuilderFactory : Ce sont des classes fournies par Spring Batch pour construire des jobs et des étapes (steps). Elles sont automatiquement injectées par Spring Boot via le contexte d'application, mais vous devez vous assurer qu'elles sont correctement incluses dans votre configuration.
* reader()**,**processor()**,**writer() : Ce sont des exemples de méthodes pour définir vos composants de lecture, de traitement et d'écriture. Vous devez vous assurer qu'ils sont correctement configurés dans votre application.

**3. Vérifier le fichier d'entrée manquant (**fichierV11.txt**) :**

L'autre erreur que vous avez rencontrée est liée à l'absence d'un fichier d'entrée (src\main\resources\input\fichierV11.txt). Assurez-vous que ce fichier existe bien à l'emplacement attendu, ou ajustez le chemin dans votre code.

Si vous avez ce fichier d'entrée sous un autre chemin, vous devez modifier le chemin dans votre PersonReader pour refléter le bon emplacement.

**4. Assurez-vous que le batch s'exécute correctement :**

Après avoir appliqué ces changements, assurez-vous que tout est bien configuré. Si vous avez des problèmes avec les étapes, vous pouvez activer le mode debug pour obtenir plus d'informations sur l'exécution du batch.

Voici comment activer le mode debug dans Spring Boot :

properties

CopierModifier

logging.level.org.springframework.batch=DEBUG

Cela vous permettra d'obtenir plus d'informations sur l'exécution des jobs et des étapes dans les logs.

**Résumé des actions à faire :**

1. Vérifiez les dépendances Spring Batch dans votre pom.xml.
2. Ajoutez la configuration Spring Batch avec les beans nécessaires (StepBuilderFactory, JobBuilderFactory, etc.).
3. Vérifiez que le fichier d'entrée est bien disponible à l'emplacement attendu.
4. Activez le mode debug pour plus d'informations sur les erreurs possibles.

Si vous avez encore des problèmes après avoir suivi ces étapes, n'hésitez pas à me donner plus de détails sur votre configuration, et je pourrai vous aider à résoudre l'erreur spécifique.

**Vous avez dit :**

je veux utiliser la version 3.4.4

**ChatGPT a dit :**

D'accord, si vous souhaitez utiliser la version 3.4.4 de Spring Batch, vous devez vous assurer que vous avez bien configuré votre projet pour cette version spécifique. Voici comment vous pouvez procéder pour résoudre le problème et configurer correctement Spring Batch version 3.4.4 dans votre projet.

**Étapes à suivre :**

1. **Mettre à jour les dépendances dans**pom.xml :  
   Assurez-vous que vous avez bien les bonnes versions de Spring Batch et Spring Boot pour votre projet.

Voici un exemple de configuration avec **Spring Batch 3.4.4** dans votre pom.xml :

xml

CopierModifier

<dependencies> *<!-- Spring Boot Starter Batch 3.4.4 -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> <version>3.4.4</version> *<!-- Utilisez la version 3.4.4 de Spring Boot -->* </dependency> *<!-- Spring Batch Core 3.4.4 -->* <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>3.4.4</version> *<!-- Utilisez la version 3.4.4 de Spring Batch -->* </dependency> *<!-- Autres dépendances nécessaires -->* *<!-- Ajouter les dépendances de votre base de données, lecteurs, etc. -->* </dependencies>

1. **Configurer**BatchConfig**pour Spring Batch 3.4.4** :  
   La configuration de Spring Batch en version 3.4.4 est similaire à ce qui est fait pour les autres versions, mais assurez-vous d'utiliser des annotations spécifiques et des méthodes adaptées à cette version.

Voici un exemple de configuration dans le fichier BatchConfig :

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.BatchConfigurer; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.Job; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing *// Active le traitement par lots dans votre application* public class BatchConfig { private final JobBuilderFactory jobBuilderFactory; private final StepBuilderFactory stepBuilderFactory; *// Injection des beans nécessaires pour construire les jobs et les steps* public BatchConfig(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory) { this.jobBuilderFactory = jobBuilderFactory; this.stepBuilderFactory = stepBuilderFactory; } @Bean public Job step1Job() { return jobBuilderFactory.get("step1Job") .start(step1()) *// Commence par la première étape* .build(); } @Bean public Step step1() { return stepBuilderFactory.get("step1") .<InputType, OutputType>chunk(10) *// Traitez des chunks de 10 éléments* .reader(reader()) *// Assurez-vous d'avoir un lecteur configuré* .processor(processor()) *// Assurez-vous d'avoir un processeur configuré* .writer(writer()) *// Assurez-vous d'avoir un writer configuré* .build(); } *// Définir les beans nécessaires pour le lecteur, le processeur et l'écrivain* public ItemReader<InputType> reader() { return new PersonReader(); *// Exemple de lecteur personnalisé* } public ItemProcessor<InputType, OutputType> processor() { return new PersonProcessor(); *// Exemple de processeur personnalisé* } public ItemWriter<OutputType> writer() { return new PersonWriter(); *// Exemple d'écrivain personnalisé* } }

**Explications :**

* + @EnableBatchProcessing : Cette annotation permet de configurer automatiquement l'infrastructure de Spring Batch (JobBuilderFactory, StepBuilderFactory, etc.) dans votre application.
  + JobBuilderFactory**et**StepBuilderFactory : Ces composants sont essentiels pour la configuration des jobs et des steps dans Spring Batch.
  + Step**et**Job : Vous construisez un job qui contient une ou plusieurs étapes.

1. **Vérification des fichiers et chemins** : Assurez-vous que le fichier d'entrée mentionné dans l'erreur (src\main\resources\input\fichierV11.txt) existe et que le chemin est correct. Si le fichier d'entrée n'existe pas, vous devrez soit le créer, soit modifier le chemin dans votre code pour qu'il corresponde au bon emplacement.
2. **Activer les logs de débogage** : Si le problème persiste, vous pouvez activer les logs de débogage pour obtenir plus d'informations sur les erreurs qui se produisent lors du démarrage de l'application. Ajoutez ceci à votre application.properties pour activer le mode débogage :

properties

CopierModifier

logging.level.org.springframework.batch=DEBUG

1. **Tester votre application** : Après avoir configuré correctement vos dépendances et vos beans, lancez votre application. Si tout est correctement configuré, votre batch devrait démarrer et traiter les données comme prévu.

**Résumé des étapes :**

1. Configurez les bonnes dépendances (spring-boot-starter-batch et spring-batch-core version 3.4.4).
2. Configurez correctement votre classe BatchConfig pour inclure les JobBuilderFactory et StepBuilderFactory.
3. Assurez-vous que votre fichier d'entrée est disponible à l'emplacement correct.
4. Activez le mode débogage pour obtenir plus d'informations en cas d'erreurs.
5. Testez l'application.

En suivant ces étapes, vous devriez pouvoir configurer et exécuter votre batch avec Spring Batch 3.4.4 sans problème. Si l'erreur persiste, donnez-moi plus de détails et je vous aiderai à trouver une solution.

**Vous avez dit :**

je veux tout les codes complet sans commentaire et que le contenu du fichier plat fichierV11.txt" et l'ecrit en json

**ChatGPT a dit :**

D'accord, voici le code complet sans commentaires et avec la transformation du contenu du fichier fichierV11.txt en JSON.

**1. Fichier**pom.xml**:**

xml

CopierModifier

<dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> <version>3.4.4</version> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>3.4.4</version> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency> </dependencies>

**2. Classe**BatchConfig.java**:**

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.BatchConfigurer; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.Job; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { private final JobBuilderFactory jobBuilderFactory; private final StepBuilderFactory stepBuilderFactory; public BatchConfig(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory) { this.jobBuilderFactory = jobBuilderFactory; this.stepBuilderFactory = stepBuilderFactory; } @Bean public Job step1Job() { return jobBuilderFactory.get("step1Job") .start(step1()) .build(); } @Bean public Step step1() { return stepBuilderFactory.get("step1") .<InputType, OutputType>chunk(10) .reader(reader()) .processor(processor()) .writer(writer()) .build(); } public ItemReader<InputType> reader() { return new PersonReader(); } public ItemProcessor<InputType, OutputType> processor() { return new PersonProcessor(); } public ItemWriter<OutputType> writer() { return new PersonWriter(); } }

**3. Classe**PersonReader.java**:**

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.item.ItemReader; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; import java.util.ArrayList; import java.util.List; public class PersonReader implements ItemReader<InputType> { private static final String FILE\_PATH = "src/main/resources/input/fichierV11.txt"; private BufferedReader reader; private List<String> lines; private int currentIndex = 0; public PersonReader() { try { reader = new BufferedReader(new FileReader(FILE\_PATH)); lines = new ArrayList<>(); String line; while ((line = reader.readLine()) != null) { lines.add(line); } } catch (IOException e) { e.printStackTrace(); } } @Override public InputType read() throws Exception { if (currentIndex < lines.size()) { String line = lines.get(currentIndex++); *// Suppose each line is a simple String, we map it to InputType* return new InputType(line); } return null; } }

**4. Classe**PersonProcessor.java**:**

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.item.ItemProcessor; import org.json.JSONObject; public class PersonProcessor implements ItemProcessor<InputType, OutputType> { @Override public OutputType process(InputType item) throws Exception { String line = item.getData(); JSONObject json = new JSONObject(); json.put("data", line); OutputType output = new OutputType(json.toString()); return output; } }

**5. Classe**PersonWriter.java**:**

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.item.ItemWriter; import java.io.FileWriter; import java.io.IOException; public class PersonWriter implements ItemWriter<OutputType> { private static final String OUTPUT\_FILE = "src/main/resources/output/output.json"; @Override public void write(List<? extends OutputType> items) throws Exception { try (FileWriter fileWriter = new FileWriter(OUTPUT\_FILE)) { for (OutputType item : items) { fileWriter.write(item.getData() + "\n"); } } catch (IOException e) { e.printStackTrace(); } } }

**6. Classe**InputType.java**:**

java

CopierModifier

package com.traitement.Traitement.config; public class InputType { private String data; public InputType(String data) { this.data = data; } public String getData() { return data; } }

**7. Classe**OutputType.java**:**

java

CopierModifier

package com.traitement.Traitement.config; public class OutputType { private String data; public OutputType(String data) { this.data = data; } public String getData() { return data; } }

**8. Fichier**fichierV11.txt**(Exemple de contenu) :**

mathematica

CopierModifier

John, Doe, 30

Jane, Smith, 25

Michael, Brown, 35

**9. Fichier de sortie JSON (**output.json**) après transformation :**

json

CopierModifier

{ "data": "John, Doe, 30" } { "data": "Jane, Smith, 25" } { "data": "Michael, Brown, 35" }

**10. Fichier**application.properties**:**

properties

CopierModifier

logging.level.org.springframework.batch=DEBUG

En suivant ces codes, le fichier fichierV11.txt est lu ligne par ligne, chaque ligne est transformée en un objet JSON, puis écrit dans un fichier de sortie sous le format JSON. Vous pouvez adapter le contenu de InputType et OutputType en fonction des données réelles dans le fichier plat.

**Vous avez dit :**

{ "nom": "John", "prenom": "Doe", "age": 25, "numCompte": "1234567890", "ville": "Paris" }, { "nom": "Jane", "prenom": "Smith", "age": 32, "numCompte": "2345678901", "ville": "Lyon" } ] le format json attendu

**ChatGPT a dit :**

D'accord, voici la mise à jour du code en tenant compte du format JSON attendu, où chaque objet contient les champs nom, prenom, age, numCompte et ville. Je vais vous montrer comment transformer les données du fichier fichierV11.txt dans ce format JSON.

**1. Exemple de contenu du fichier**fichierV11.txt**:**

plaintext

CopierModifier

John,Doe,25,1234567890,Paris Jane,Smith,32,2345678901,Lyon

**2. Classe**PersonReader.java**: (Lecture du fichier**fichierV11.txt**et transformation en objets)**

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.item.ItemReader; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; import java.util.ArrayList; import java.util.List; public class PersonReader implements ItemReader<InputType> { private static final String FILE\_PATH = "src/main/resources/input/fichierV11.txt"; private BufferedReader reader; private List<String> lines; private int currentIndex = 0; public PersonReader() { try { reader = new BufferedReader(new FileReader(FILE\_PATH)); lines = new ArrayList<>(); String line; while ((line = reader.readLine()) != null) { lines.add(line); } } catch (IOException e) { e.printStackTrace(); } } @Override public InputType read() throws Exception { if (currentIndex < lines.size()) { String line = lines.get(currentIndex++); String[] parts = line.split(","); *// Assumons que le format est : nom, prénom, age, numCompte, ville* return new InputType(parts[0], parts[1], Integer.parseInt(parts[2]), parts[3], parts[4]); } return null; } }

**3. Classe**InputType.java**:**

java

CopierModifier

package com.traitement.Traitement.config; public class InputType { private String nom; private String prenom; private int age; private String numCompte; private String ville; public InputType(String nom, String prenom, int age, String numCompte, String ville) { this.nom = nom; this.prenom = prenom; this.age = age; this.numCompte = numCompte; this.ville = ville; } public String getNom() { return nom; } public String getPrenom() { return prenom; } public int getAge() { return age; } public String getNumCompte() { return numCompte; } public String getVille() { return ville; } }

**4. \*\*Classe**

**Vous avez dit :**

contenu du fichierV11.txt Neha Amina 51 1029 Casablanca Qasimi Youssef 58 548 Tanger Hamid Hafsa 25 370 Fès Neha Khalid 52 1095 Casablanca Benjelloun Fatima 58 452 Alexandrie Hiba Fatima 43 366 Marrakech Neha Omar 55 764 Le Caire Mansour Youssef 40 1097 Alexandrie Hamid Leila 49 721 Tunis Mansour Leila 23 952 Marrakech Bouazza Ahmed 35 223 Dubaï Zahir Hafsa 48 647 Marrakech Oualid Ahmed 26 710 Marrakech Cherkaoui Omar 36 499 Rabat Cherkaoui Youssef 43 573 Casablanca Hiba Amina 46 448 Marrakech Bouazza Leila 52 794 Fès Alami Omar 55 231 Le Caire Zahir Fatima 49 784 Rabat Cherkaoui Mohamed 42 707 Dubaï Benjelloun Zahra 30 726 Casablanca Cherkaoui Amina 32 382 Le Caire Idrissi Omar 35 429 Tunis Neha Youssef 20 744 Fès Neha Ahmed 59 697 Le Caire Bouazza Khalid 47 898 Casablanca Oualid Mohamed 56 619 Dubaï Bouazza Fatima 20 666 Dubaï Hiba Leila 24 789 Casablanca Hamid Omar 56 435 Tanger Hamid Mohamed 58 703 Casablanca Alami Khalid 34 193 Casablanca Zahir Hafsa 30 257 Dubaï Hamid Khalid 27 525 Casablanca Zahir Youssef 43 777 Tunis Idrissi Omar 44 698 Rabat Saadi Hafsa 26 501 Le Caire Zahir Khalid 24 795 Tunis Hamid Khalid 52 1023 Casablanca Oualid Omar 35 836 Le Caire Alami Youssef 53 157 Alexandrie Qasimi Mohamed 35 101 Dubaï Hiba Youssef 24 970 Casablanca Hiba Hafsa 22 131 Riyad Zahir Mohamed 35 654 Dubaï Saadi Fatima 39 555 Riyad Cherkaoui Hafsa 20 320 Riyad Neha Amina 29 145 Rabat Zahir Omar 35 120 Riyad Bouazza Ahmed 59 691 Le Caire Alami Mohamed 35 1030 Rabat Alami Fatima 33 321 Riyad Bouazza Leila 42 537 Fès Hamid Amina 34 944 Tanger Neha Khalid 58 824 Tunis Mansour Fatima 34 564 Marrakech Neha Khalid 29 496 Rabat Hamid Zahra 22 757 Riyad Oualid Mohamed 21 367 Alexandrie Bouazza Fatima 50 248 Marrakech Cherkaoui Youssef 53 550 Alexandrie Neha Youssef 55 431 Rabat Benjelloun Mohamed 25 521 Le Caire Hiba Khalid 41 224 Le Caire Mansour Fatima 20 621 Rabat Mansour Fatima 38 181 Marrakech Cherkaoui Ahmed 49 930 Casablanca Hiba Leila 34 252 Rabat Hiba Mohamed 41 891 Dubaï Idrissi Mohamed 42 828 Casablanca Bouazza Amina 34 998 Le Caire Mansour Mohamed 44 600 Alexandrie Idrissi Ahmed 24 414 Alexandrie Hamid Leila 38 958 Le Caire Hamid Ahmed 38 176 Dubaï Idrissi Mohamed 49 420 Fès Mansour Amina 53 486 Tanger Hamid Amina 40 842 Le Caire Zahir Khalid 54 1013 Riyad Qasimi Omar 49 790 Tunis Cherkaoui Ahmed 55 860 Tunis Saadi Leila 24 894 Tanger Qasimi Ahmed 40 399 Casablanca Alami Mohamed 43 267 Riyad El Fassi Ahmed 36 438 Fès Benjelloun Amina 24 214 Tanger Mansour Khalid 57 806 Tanger Qasimi Omar 53 333 Alexandrie Qasimi Fatima 30 745 Alexandrie El Fassi Mohamed 31 121 Le Caire Saadi Fatima 25 766 Tanger Qasimi Leila 32 846 Le Caire Bouazza Amina 28 115 Riyad Bouazza Mohamed 42 979 Marrakech Hamid Hafsa 35 546 Alexandrie Oualid Ahmed 37 895 Dubaï Saadi Youssef 47 263 Casablanca Idrissi Fatima 25 118 Fès Alami Youssef 37 536 Dubaï Hiba Zahra 57 575 Fès Zahir Ahmed 27 507 Alexandrie Zahir Zahra 30 217 Alexandrie Hamid Fatima 60 545 Tunis Bouazza Ahmed 45 133 Marrakech Mansour Omar 45 1043 Tunis Neha Amina 20 505 Fès Cherkaoui Omar 42 472 Rabat Hamid Zahra 36 271 Fès Benjelloun Mohamed 60 189 Alexandrie Oualid Omar 59 791 Alexandrie Alami Mohamed 44 904 Casablanca Bouazza Hafsa 25 785 Fès Mansour Khalid 35 996 Tanger Neha Fatima 44 289 Tanger Idrissi Hafsa 59 798 Tanger Bouazza Leila 28 1024 Fès Qasimi Ahmed 44 1052 Casablanca El Fassi Ahmed 35 132 Fès Alami Ahmed 52 816 Casablanca Bouazza Zahra 51 361 Tanger Saadi Ahmed 48 377 Rabat Alami Mohamed 27 407 Casablanca Saadi Ahmed 50 823 Rabat Idrissi Youssef 34 643 Casablanca Cherkaoui Youssef 41 344 Tanger Oualid Fatima 47 592 Rabat El Fassi Amina 33 831 Marrakech Oualid Ahmed 27 322 Marrakech Zahir Omar 60 735 Tanger Oualid Amina 27 937 Fès Alami Amina 30 663 Rabat Cherkaoui Omar 22 922 Le Caire Zahir Zahra 53 713 Le Caire Saadi Ahmed 29 668 Dubaï Bouazza Mohamed 32 731 Tanger Hamid Youssef 48 569 Tunis Mansour Leila 24 515 Rabat Cherkaoui Khalid 22 687 Dubaï Alami Mohamed 46 574 Riyad Cherkaoui Zahra 27 112 Marrakech Bouazza Mohamed 52 237 Tanger Hiba Khalid 45 139 Rabat Bouazza Omar 56 1049 Riyad Neha Amina 33 191 Marrakech Alami Hafsa 55 653 Dubaï Mansour Khalid 55 992 Dubaï Saadi Mohamed 39 617 Alexandrie Neha Hafsa 37 834 Alexandrie Bouazza Youssef 48 916 Rabat Saadi Hafsa 28 675 Tanger Neha Hafsa 59 803 Rabat Benjelloun Hafsa 57 209 Rabat Saadi Zahra 50 445 Riyad Qasimi Leila 56 758 Riyad Hamid Leila 60 100 Marrakech Hiba Ahmed 60 702 Marrakech Neha Khalid 31 142 Casablanca Zahir Amina 29 510 Tanger Neha Leila 41 727 Tanger Saadi Amina 52 457 Casablanca Benjelloun Hafsa 28 670 Tunis Benjelloun Mohamed 28 630 Marrakech Hiba Zahra 32 218 Le Caire Neha Zahra 25 1069 Rabat Idrissi Omar 50 512 Riyad Oualid Amina 32 925 Dubaï Hiba Fatima 48 146 Tanger Hiba Leila 27 287 Fès Benjelloun Ahmed 20 781 Casablanca Cherkaoui Khalid 43 974 Rabat Zahir Khalid 35 724 Tunis Alami Omar 46 625 Tanger Hamid Hafsa 40 200 Dubaï Bouazza Leila 48 273 Marrakech Idrissi Mohamed 24 869 Tunis Hamid Amina 24 801 Dubaï Hamid Hafsa 26 917 Dubaï Saadi Fatima 25 631 Marrakech Alami Leila 38 123 Tanger Benjelloun Khalid 57 1087 Dubaï Saadi Youssef 24 342 Casablanca Hiba Mohamed 20 942 Tanger Bouazza Khalid 52 720 Casablanca Neha Fatima 35 883 Riyad Hamid Omar 39 760 Tunis Zahir Omar 40 805 Dubaï Saadi Zahra 60 467 Tunis Oualid Leila 28 415 Alexandrie Benjelloun Hafsa 37 570 Tanger Alami Omar 20 1062 Tunis Cherkaoui Amina 31 396 Riyad Oualid Leila 20 230 Tanger Neha Leila 59 504 Riyad Qasimi Omar 31 162 Rabat Zahir Leila 45 542 Le Caire Alami Youssef 58 130 Le Caire Mansour Hafsa 45 577 Dubaï Oualid Ahmed 36 830 Dubaï Idrissi Zahra 28 516 Fès Mansour Mohamed 56 676 Rabat Qasimi Youssef 42 603 Le Caire Alami Amina 58 855 Riyad Cherkaoui Mohamed 60 755 Riyad Qasimi Amina 51 222 Tanger Zahir Omar 48 313 Riyad Alami Ahmed 54 911 Tanger Zahir Youssef 32 941 Fès Qasimi Youssef 44 955 Marrakech Idrissi Zahra 26 354 Alexandrie El Fassi Zahra 34 580 Tanger Hamid Hafsa 44 179 Casablanca Alami Omar 44 786 Le Caire El Fassi Mohamed 51 993 Rabat Neha Ahmed 29 1000 Alexandrie El Fassi Youssef 54 618 Fès Alami Amina 46 747 Marrakech Benjelloun Ahmed 59 780 Alexandrie Oualid Zahra 39 427 Tanger Zahir Omar 23 395 Rabat Oualid Mohamed 23 155 Tunis Oualid Ahmed 42 964 Marrakech Neha Ahmed 55 708 Tunis Hamid Zahra 29 330 Fès Mansour Mohamed 54 314 Casablanca Cherkaoui Youssef 47 318 Fès El Fassi Youssef 38 865 Le Caire Mansour Zahra 36 797 Dubaï Mansour Amina 29 915 Le Caire Benjelloun Ahmed 29 245 Marrakech Zahir Khalid 50 596 Marrakech El Fassi Zahra 30 173 Dubaï Oualid Fatima 41 1033 Tunis Idrissi Hafsa 46 409 Le Caire Alami Youssef 48 153 Marrakech Benjelloun Amina 27 116 Riyad Bouazza Leila 49 274 Tunis Zahir Omar 54 931 Marrakech Oualid Amina 37 1072 Fès Saadi Hafsa 24 442 Fès Benjelloun Khalid 32 919 Tunis Qasimi Khalid 49 556 Dubaï Neha Fatima 49 114 Rabat Qasimi Omar 28 349 Le Caire Qasimi Youssef 40 345 Rabat Saadi Fatima 27 400 Alexandrie Hamid Leila 60 954 Rabat Saadi Zahra 39 171 Rabat Qasimi Youssef 21 584 Dubaï Hiba Ahmed 36 939 Rabat Mansour Hafsa 35 532 Alexandrie Hiba Amina 60 560 Fès Saadi Khalid 38 166 Dubaï Saadi Hafsa 23 887 Riyad Hamid Youssef 56 723 Casablanca Mansour Fatima 36 934 Marrakech Idrissi Ahmed 40 335 Marrakech El Fassi Ahmed 44 302 Tanger Oualid Omar 60 109 Riyad Saadi Khalid 48 819 Tunis Hamid Youssef 24 770 Le Caire Cherkaoui Ahmed 46 826 Rabat Hamid Amina 48 658 Tunis Zahir Leila 41 362 Le Caire El Fassi Hafsa 35 1019 Tanger Idrissi Amina 33 439 Le Caire Benjelloun Leila 26 880 Fès Benjelloun Mohamed 48 1035 Le Caire Hamid Khalid 33 529 Marrakech Zahir Youssef 29 748 Alexandrie Zahir Youssef 22 642 Dubaï Bouazza Amina 41 1005 Marrakech Idrissi Amina 37 485 Fès Alami Youssef 52 283 Marrakech El Fassi Mohamed 41 315 Le Caire Mansour Youssef 23 614 Fès Benjelloun Amina 22 494 Tunis Saadi Ahmed 58 686 Le Caire Hamid Mohamed 29 432 Casablanca Zahir Fatima 36 151 Riyad Saadi Youssef 44 468 Marrakech Saadi Hafsa 43 655 Casablanca Neha Youssef 42 408 Fès Alami Ahmed 26 343 Riyad Oualid Mohamed 49 962 Casablanca Hamid Zahra 48 355 Casablanca Saadi Fatima 28 497 Alexandrie El Fassi Omar 37 893 Marrakech Hiba Ahmed 20 397 Alexandrie Qasimi Youssef 25 850 Alexandrie Alami Mohamed 24 890 Riyad Oualid Omar 47 125 Alexandrie Idrissi Youssef 30 402 Tanger Idrissi Ahmed 49 319 Riyad Benjelloun Omar 39 480 Alexandrie Benjelloun Hafsa 45 316 Fès Hamid Omar 20 606 Le Caire Zahir Youssef 48 1038 Alexandrie Saadi Mohamed 35 980 Riyad Qasimi Leila 57 360 Dubaï Saadi Hafsa 56 549 Fès Zahir Mohamed 21 364 Le Caire Zahir Zahra 40 462 Marrakech Mansour Mohamed 37 1016 Alexandrie Idrissi Mohamed 37 426 Rabat Hiba Mohamed 54 693 Tunis Cherkaoui Fatima 45 444 Dubaï Alami Amina 23 220 Fès Mansour Omar 27 807 Marrakech Bouazza Omar 60 629 Le Caire Hiba Zahra 26 1039 Marrakech Hamid Mohamed 50 352 Fès Hiba Youssef 47 1090 Fès Hamid Youssef 46 749 Casablanca Saadi Khalid 32 381 Marrakech Zahir Mohamed 44 1028 Tanger Cherkaoui Fatima 27 428 Le Caire Cherkaoui Fatima 45 483 Fès Idrissi Amina 25 524 Dubaï Bouazza Youssef 38 339 Fès

**ChatGPT a dit :**

D'accord, avec le contenu du fichier fichierV11.txt que vous m'avez fourni, je vais maintenant vous donner un code Java qui lit chaque ligne du fichier, traite les données et les écrit en format JSON avec les champs attendus : nom, prenom, age, numCompte, et ville.

**1. Classe**PersonReader.java**: (Lecture du fichier et création d'objets)**

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.item.ItemReader; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; import java.util.ArrayList; import java.util.List; public class PersonReader implements ItemReader<InputType> { private static final String FILE\_PATH = "src/main/resources/input/fichierV11.txt"; private BufferedReader reader; private List<String> lines; private int currentIndex = 0; public PersonReader() { try { reader = new BufferedReader(new FileReader(FILE\_PATH)); lines = new ArrayList<>(); String line; while ((line = reader.readLine()) != null) { lines.add(line); } } catch (IOException e) { e.printStackTrace(); } } @Override public InputType read() throws Exception { if (currentIndex < lines.size()) { String line = lines.get(currentIndex++); String[] parts = line.split("\\s+"); *// Assumons que le format est : nom, prénom, age, numCompte, ville* return new InputType(parts[0], parts[1], Integer.parseInt(parts[2]), parts[3], parts[4]); } return null; } }

**2. Classe**InputType.java**: (Modélisation des données)**

java

CopierModifier

package com.traitement.Traitement.config; public class InputType { private String nom; private String prenom; private int age; private String numCompte; private String ville; public InputType(String nom, String prenom, int age, String numCompte, String ville) { this.nom = nom; this.prenom = prenom; this.age = age; this.numCompte = numCompte; this.ville = ville; } public String getNom() { return nom; } public String getPrenom() { return prenom; } public int getAge() { return age; } public String getNumCompte() { return numCompte; } public String getVille() { return ville; } }

**3. Classe**PersonProcessor.java**: (Transformation en format JSON)**

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.item.ItemProcessor; import org.json.JSONObject; public class PersonProcessor implements ItemProcessor<InputType, String> { @Override public String process(InputType input) throws Exception { JSONObject jsonObject = new JSONObject(); jsonObject.put("nom", input.getNom()); jsonObject.put("prenom", input.getPrenom()); jsonObject.put("age", input.getAge()); jsonObject.put("numCompte", input.getNumCompte()); jsonObject.put("ville", input.getVille()); return jsonObject.toString(); } }

**4. Classe**JsonWriter.java**: (Écriture du JSON dans un fichier)**

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.item.ItemWriter; import java.io.BufferedWriter; import java.io.FileWriter; import java.io.IOException; import java.util.List; public class JsonWriter implements ItemWriter<String> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/output.json"; @Override public void write(List<? extends String> items) throws Exception { try (BufferedWriter writer = new BufferedWriter(new FileWriter(OUTPUT\_FILE\_PATH))) { writer.write("[\n"); for (int i = 0; i < items.size(); i++) { writer.write(items.get(i)); if (i < items.size() - 1) { writer.write(",\n"); } } writer.write("\n]"); } catch (IOException e) { e.printStackTrace(); } } }

**5. Configuration Spring Batch**

Dans votre fichier de configuration Spring Batch, vous allez lier ces classes ensemble pour que le traitement du fichier fonctionne.

java

CopierModifier

package com.traitement.Traitement.config; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.batch.item.support.ListItemWriter; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public Step step1(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("step1") .<InputType, String>chunk(10) .reader(new PersonReader()) .processor(new PersonProcessor()) .writer(new JsonWriter()) .build(); } @Bean public Job job(JobBuilderFactory jobBuilderFactory, Step step1) { return jobBuilderFactory.get("job") .incrementer(new RunIdIncrementer()) .flow(step1) .end() .build(); } }

**6. Exécution et résultat**

Le fichier fichierV11.txt que vous m'avez donné sera lu et chaque ligne sera convertie en un objet JSON au format suivant :

json

CopierModifier

[ { "nom": "Neha", "prenom": "Amina", "age": 51, "numCompte": "1029", "ville": "Casablanca" }, { "nom": "Qasimi", "prenom": "Youssef", "age": 58, "numCompte": "548", "ville": "Tanger" }, ... ]

**Remarque:**

* Ce code suppose que chaque champ dans fichierV11.txt est séparé par des espaces multiples (\\s+ dans l'expression régulière).
* Le fichier JSON généré sera stocké dans le fichier output.json dans le répertoire src/main/resources/output/.

**Vous avez dit :**

String[] parts = line.split("\\s+"); // Assumons que le format est : nom, prénom, age, numCompte, ville return new InputType(parts[0], parts[1], Integer.parseInt(parts[2]), parts[3], parts[4]); } le format n'est pas comme ca mais on travail avecc les position Neha Amina 51 1029 Casablanca Qasimi Youssef 58 548 Tanger Hamid Hafsa 25 370 Fès Neha Khalid 52 1095 Casablanca Benjelloun Fatima 58 452 Alexandrie Hiba Fatima 43 366 Marrakech Neha Omar 55 764 Le Caire Mansour Youssef 40 1097 Alexandrie Hamid Leila 49 721 Tunis Mansour Leila 23 952 Marrakech Bouazza Ahmed 35 223 Dubaï Zahir Hafsa 48 647 Marrakech Oualid Ahmed 26 710 Marrakech Cherkaoui Omar 36 499 Rabat Cherkaoui Youssef 43 573 Casablanca Hiba Amina 46 448 Marrakech Bouazza Leila 52 794 Fès Alami Omar 55 231 Le Caire Zahir Fatima 49 784 Rabat Cherkaoui Mohamed 42 707 Dubaï Benjelloun Zahra 30 726 Casablanca Cherkaoui Amina 32 382 Le Caire Idrissi Omar 35 429 Tunis Neha Youssef 20 744 Fès Neha Ahmed 59 697 Le Caire Bouazza Khalid 47 898 Casablanca Oualid Mohamed 56 619 Dubaï Bouazza Fatima 20 666 Dubaï Hiba Leila 24 789 Casablanca Hamid Omar 56 435 Tanger Hamid Mohamed 58 703 Casablanca Alami Khalid 34 193 Casablanca Zahir Hafsa 30 257 Dubaï Hamid Khalid 27 525 Casablanca Zahir Youssef 43 777 Tunis Idrissi Omar 44 698 Rabat Saadi Hafsa 26 501 Le Caire Zahir Khalid 24 795 Tunis Hamid Khalid 52 1023 Casablanca Oualid Omar 35 836 Le Caire Alami Youssef 53 157 Alexandrie Qasimi Mohamed 35 101 Dubaï Hiba Youssef 24 970 Casablanca Hiba Hafsa 22 131 Riyad Zahir Mohamed 35 654 Dubaï Saadi Fatima 39 555 Riyad Cherkaoui Hafsa 20 320 Riyad Neha Amina 29 145 Rabat Zahir Omar 35 120 Riyad Bouazza Ahmed 59 691 Le Caire Alami Mohamed 35 1030 Rabat Alami Fatima 33 321 Riyad Bouazza Leila 42 537 Fès Hamid Amina 34 944 Tanger Neha Khalid 58 824 Tunis Mansour Fatima 34 564 Marrakech Neha Khalid 29 496 Rabat Hamid Zahra 22 757 Riyad Oualid Mohamed 21 367 Alexandrie Bouazza Fatima 50 248 Marrakech Cherkaoui Youssef 53 550 Alexandrie Neha Youssef 55 431 Rabat Benjelloun Mohamed 25 521 Le Caire Hiba Khalid 41 224 Le Caire Mansour Fatima 20 621 Rabat Mansour Fatima 38 181 Marrakech Cherkaoui Ahmed 49 930 Casablanca Hiba Leila 34 252 Rabat Hiba Mohamed 41 891 Dubaï Idrissi Mohamed 42 828 Casablanca Bouazza Amina 34 998 Le Caire Mansour Mohamed 44 600 Alexandrie Idrissi Ahmed 24 414 Alexandrie Hamid Leila 38 958 Le Caire Hamid Ahmed 38 176 Dubaï Idrissi Mohamed 49 420 Fès Mansour Amina 53 486 Tanger Hamid Amina 40 842 Le Caire Zahir Khalid 54 1013 Riyad Qasimi Omar 49 790 Tunis Cherkaoui Ahmed 55 860 Tunis Saadi Leila 24 894 Tanger Qasimi Ahmed 40 399 Casablanca Alami Mohamed 43 267 Riyad El Fassi Ahmed 36 438 Fès Benjelloun Amina 24 214 Tanger Mansour Khalid 57 806 Tanger Qasimi Omar 53 333 Alexandrie Qasimi Fatima 30 745 Alexandrie El Fassi Mohamed 31 121 Le Caire Saadi Fatima 25 766 Tanger Qasimi Leila 32 846 Le Caire Bouazza Amina 28 115 Riyad Bouazza Mohamed 42 979 Marrakech Hamid Hafsa 35 546 Alexandrie Oualid Ahmed 37 895 Dubaï Saadi Youssef 47 263 Casablanca Idrissi Fatima 25 118 Fès Alami Youssef 37 536 Dubaï Hiba Zahra 57 575 Fès Zahir Ahmed 27 507 Alexandrie Zahir Zahra 30 217 Alexandrie Hamid Fatima 60 545 Tunis Bouazza Ahmed 45 133 Marrakech Mansour Omar 45 1043 Tunis Neha Amina 20 505 Fès Cherkaoui Omar 42 472 Rabat Hamid Zahra 36 271 Fès Benjelloun Mohamed 60 189 Alexandrie Oualid Omar 59 791 Alexandrie Alami Mohamed 44 904 Casablanca Bouazza Hafsa 25 785 Fès Mansour Khalid 35 996 Tanger Neha Fatima 44 289 Tanger Idrissi Hafsa 59 798 Tanger Bouazza Leila 28 1024 Fès Qasimi Ahmed 44 1052 Casablanca El Fassi Ahmed 35 132 Fès Alami Ahmed 52 816 Casablanca Bouazza Zahra 51 361 Tanger Saadi Ahmed 48 377 Rabat Alami Mohamed 27 407 Casablanca Saadi Ahmed 50 823 Rabat Idrissi Youssef 34 643 Casablanca Cherkaoui Youssef 41 344 Tanger Oualid Fatima 47 592 Rabat El Fassi Amina 33 831 Marrakech Oualid Ahmed 27 322 Marrakech Zahir Omar 60 735 Tanger Oualid Amina 27 937 Fès Alami Amina 30 663 Rabat Cherkaoui Omar 22 922 Le Caire Zahir Zahra 53 713 Le Caire Saadi Ahmed 29 668 Dubaï Bouazza Mohamed 32 731 Tanger Hamid Youssef 48 569 Tunis Mansour Leila 24 515 Rabat Cherkaoui Khalid 22 687 Dubaï Alami Mohamed 46 574 Riyad Cherkaoui Zahra 27 112 Marrakech Bouazza Mohamed 52 237 Tanger Hiba Khalid 45 139 Rabat Bouazza Omar 56 1049 Riyad Neha Amina 33 191 Marrakech Alami Hafsa 55 653 Dubaï Mansour Khalid 55 992 Dubaï Saadi Mohamed 39 617 Alexandrie Neha Hafsa 37 834 Alexandrie Bouazza Youssef 48 916 Rabat Saadi Hafsa 28 675 Tanger Neha Hafsa 59 803 Rabat Benjelloun Hafsa 57 209 Rabat Saadi Zahra 50 445 Riyad Qasimi Leila 56 758 Riyad Hamid Leila 60 100 Marrakech Hiba Ahmed 60 702 Marrakech Neha Khalid 31 142 Casablanca Zahir Amina 29 510 Tanger Neha Leila 41 727 Tanger Saadi Amina 52 457 Casablanca Benjelloun Hafsa 28 670 Tunis Benjelloun Mohamed 28 630 Marrakech Hiba Zahra 32 218 Le Caire Neha Zahra 25 1069 Rabat Idrissi Omar 50 512 Riyad Oualid Amina 32 925 Dubaï Hiba Fatima 48 146 Tanger Hiba Leila 27 287 Fès Benjelloun Ahmed 20 781 Casablanca Cherkaoui Khalid 43 974 Rabat Zahir Khalid 35 724 Tunis Alami Omar 46 625 Tanger Hamid Hafsa 40 200 Dubaï Bouazza Leila 48 273 Marrakech Idrissi Mohamed 24 869 Tunis Hamid Amina 24 801 Dubaï Hamid Hafsa 26 917 Dubaï Saadi Fatima 25 631 Marrakech Alami Leila 38 123 Tanger Benjelloun Khalid 57 1087 Dubaï Saadi Youssef 24 342 Casablanca Hiba Mohamed 20 942 Tanger Bouazza Khalid 52 720 Casablanca Neha Fatima 35 883 Riyad Hamid Omar 39 760 Tunis Zahir Omar 40 805 Dubaï Saadi Zahra 60 467 Tunis Oualid Leila 28 415 Alexandrie Benjelloun Hafsa 37 570 Tanger Alami Omar 20 1062 Tunis Cherkaoui Amina 31 396 Riyad Oualid Leila 20 230 Tanger Neha Leila 59 504 Riyad Qasimi Omar 31 162 Rabat Zahir Leila 45 542 Le Caire Alami Youssef 58 130 Le Caire Mansour Hafsa 45 577 Dubaï Oualid Ahmed 36 830 Dubaï Idrissi Zahra 28 516 Fès Mansour Mohamed 56 676 Rabat Qasimi Youssef 42 603 Le Caire Alami Amina 58 855 Riyad Cherkaoui Mohamed 60 755 Riyad Qasimi Amina 51 222 Tanger Zahir Omar 48 313 Riyad Alami Ahmed 54 911 Tanger Zahir Youssef 32 941 Fès Qasimi Youssef 44 955 Marrakech Idrissi Zahra 26 354 Alexandrie El Fassi Zahra 34 580 Tanger Hamid Hafsa 44 179 Casablanca Alami Omar 44 786 Le Caire El Fassi Mohamed 51 993 Rabat Neha Ahmed 29 1000 Alexandrie El Fassi Youssef 54 618 Fès Alami Amina 46 747 Marrakech Benjelloun Ahmed 59 780 Alexandrie Oualid Zahra 39 427 Tanger Zahir Omar 23 395 Rabat Oualid Mohamed 23 155 Tunis Oualid Ahmed 42 964 Marrakech Neha Ahmed 55 708 Tunis Hamid Zahra 29 330 Fès Mansour Mohamed 54 314 Casablanca Cherkaoui Youssef 47 318 Fès El Fassi Youssef 38 865 Le Caire Mansour Zahra 36 797 Dubaï Mansour Amina 29 915 Le Caire Benjelloun Ahmed 29 245 Marrakech Zahir Khalid 50 596 Marrakech El Fassi Zahra 30 173 Dubaï Oualid Fatima 41 1033 Tunis Idrissi Hafsa 46 409 Le Caire Alami Youssef 48 153 Marrakech Benjelloun Amina 27 116 Riyad Bouazza Leila 49 274 Tunis Zahir Omar 54 931 Marrakech Oualid Amina 37 1072 Fès Saadi Hafsa 24 442 Fès Benjelloun Khalid 32 919 Tunis Qasimi Khalid 49 556 Dubaï Neha Fatima 49 114 Rabat Qasimi Omar 28 349 Le Caire Qasimi Youssef 40 345 Rabat Saadi Fatima 27 400 Alexandrie Hamid Leila 60 954 Rabat Saadi Zahra 39 171 Rabat Qasimi Youssef 21 584 Dubaï Hiba Ahmed 36 939 Rabat Mansour Hafsa 35 532 Alexandrie Hiba Amina 60 560 Fès Saadi Khalid 38 166 Dubaï Saadi Hafsa 23 887 Riyad Hamid Youssef 56 723 Casablanca Mansour Fatima 36 934 Marrakech Idrissi Ahmed 40 335 Marrakech El Fassi Ahmed 44 302 Tanger Oualid Omar 60 109 Riyad Saadi Khalid 48 819 Tunis Hamid Youssef 24 770 Le Caire Cherkaoui Ahmed 46 826 Rabat Hamid Amina 48 658 Tunis Zahir Leila 41 362 Le Caire El Fassi Hafsa 35 1019 Tanger Idrissi Amina 33 439 Le Caire Benjelloun Leila 26 880 Fès Benjelloun Mohamed 48 1035 Le Caire Hamid Khalid 33 529 Marrakech Zahir Youssef 29 748 Alexandrie Zahir Youssef 22 642 Dubaï Bouazza Amina 41 1005 Marrakech Idrissi Amina 37 485 Fès Alami Youssef 52 283 Marrakech El Fassi Mohamed 41 315 Le Caire Mansour Youssef 23 614 Fès Benjelloun Amina 22 494 Tunis Saadi Ahmed 58 686 Le Caire Hamid Mohamed 29 432 Casablanca Zahir Fatima 36 151 Riyad Saadi Youssef 44 468 Marrakech Saadi Hafsa 43 655 Casablanca Neha Youssef 42 408 Fès Alami Ahmed 26 343 Riyad Oualid Mohamed 49 962 Casablanca Hamid Zahra 48 355 Casablanca Saadi Fatima 28 497 Alexandrie El Fassi Omar 37 893 Marrakech Hiba Ahmed 20 397 Alexandrie Qasimi Youssef 25 850 Alexandrie Alami Mohamed 24 890 Riyad Oualid Omar 47 125 Alexandrie Idrissi Youssef 30 402 Tanger Idrissi Ahmed 49 319 Riyad Benjelloun Omar 39 480 Alexandrie Benjelloun Hafsa 45 316 Fès Hamid Omar 20 606 Le Caire Zahir Youssef 48 1038 Alexandrie Saadi Mohamed 35 980 Riyad Qasimi Leila 57 360 Dubaï Saadi Hafsa 56 549 Fès Zahir Mohamed 21 364 Le Caire Zahir Zahra 40 462 Marrakech Mansour Mohamed 37 1016 Alexandrie Idrissi Mohamed 37 426 Rabat Hiba Mohamed 54 693 Tunis Cherkaoui Fatima 45 444 Dubaï Alami Amina 23 220 Fès Mansour Omar 27 807 Marrakech Bouazza Omar 60 629 Le Caire Hiba Zahra 26 1039 Marrakech Hamid Mohamed 50 352 Fès Hiba Youssef 47 1090 Fès Hamid Youssef 46 749 Casablanca Saadi Khalid 32 381 Marrakech Zahir Mohamed 44 1028 Tanger Cherkaoui Fatima 27 428 Le Caire Cherkaoui Fatima 45 483 Fès Idrissi Amina 25 524 Dubaï Bouazza Youssef 38 339 Fès

**ChatGPT a dit :**

Pour adapter votre code à ce format de données où les valeurs sont réparties par position, vous pouvez procéder comme suit :

**Exemple de code Java qui fonctionne avec ce format :**

java

CopierModifier

public InputType parseLine(String line) { *// Utilisation de la méthode split pour diviser la ligne en différentes parties, basées sur des espaces* String[] parts = line.split("\\s+"); *// On suppose que chaque ligne suit ce modèle : prénom, nom, âge, numéro de compte, ville* String prenom = parts[0]; *// 1ère colonne - prénom* String nom = parts[1]; *// 2ème colonne - nom* int age = Integer.parseInt(parts[2]); *// 3ème colonne - âge* String numCompte = parts[3]; *// 4ème colonne - numéro de compte* String ville = parts[4]; *// 5ème colonne - ville* *// Retour d'un nouvel objet InputType avec les valeurs extraites* return new InputType(prenom, nom, age, numCompte, ville); }

**Explication :**

1. **Séparation des données** : Nous utilisons la méthode split("\\s+"), qui découpe chaque ligne en plusieurs sous-chaînes selon les espaces. Cette expression régulière permet de gérer les espaces multiples entre les éléments.
2. **Accès aux colonnes** : Vous accédez ensuite directement à chaque élément en fonction de sa position (par exemple, parts[0] pour le prénom, parts[1] pour le nom, etc.).
3. **Conversion de l'âge** : Nous convertissons l'âge en entier avec Integer.parseInt(parts[2]), en supposant que l'âge est toujours un nombre valide dans la chaîne.
4. **Création de l'objet**InputType : Enfin, un nouvel objet de type InputType est créé avec les données extraites de la ligne et est retourné.

**Exemple de données traitées :**

Avec une ligne comme :

yaml

CopierModifier

Neha Amina 51 1029 Casablanca

Le code extraira :

* **Prenom** : Neha
* **Nom** : Amina
* **Age** : 51
* **NumCompte** : 1029
* **Ville** : Casablanca

Ainsi, cela vous permet d'extraire les informations selon leur position dans la ligne, comme spécifié.

Cela fonctionne bien tant que les lignes ont toujours le même format (c'est-à-dire 5 éléments séparés par des espaces).

**Vous avez dit :**

s'il te plait rend moi un code complet qui fait ce que j'ai demander qui prend le fichierV11 Neha Amina 51 1029 Casablanca Qasimi Youssef 58 548 Tanger Hamid Hafsa 25 370 Fès Neha Khalid 52 1095 Casablanca Benjelloun Fatima 58 452 Alexandrie Hiba Fatima 43 366 Marrakech Neha Omar 55 764 Le Caire Mansour Youssef 40 1097 Alexandrie Hamid Leila 49 721 Tunis Mansour Leila 23 952 Marrakech Bouazza Ahmed 35 223 Dubaï Zahir Hafsa 48 647 Marrakech Oualid Ahmed 26 710 Marrakech Cherkaoui Omar 36 499 Rabat Cherkaoui Youssef 43 573 Casablanca Hiba Amina 46 448 Marrakech Bouazza Leila 52 794 Fès Alami Omar 55 231 Le Caire Zahir Fatima 49 784 Rabat Cherkaoui Mohamed 42 707 Dubaï Benjelloun Zahra 30 726 Casablanca Cherkaoui Amina 32 382 Le Caire Idrissi Omar 35 429 Tunis Neha Youssef 20 744 Fès Neha Ahmed 59 697 Le Caire Bouazza Khalid 47 898 Casablanca Oualid Mohamed 56 619 Dubaï Bouazza Fatima 20 666 Dubaï Hiba Leila 24 789 Casablanca Hamid Omar 56 435 Tanger Hamid Mohamed 58 703 Casablanca Alami Khalid 34 193 Casablanca Zahir Hafsa 30 257 Dubaï Hamid Khalid 27 525 Casablanca Zahir Youssef 43 777 Tunis Idrissi Omar 44 698 Rabat Saadi Hafsa 26 501 Le Caire Zahir Khalid 24 795 Tunis Hamid Khalid 52 1023 Casablanca Oualid Omar 35 836 Le Caire Alami Youssef 53 157 Alexandrie Qasimi Mohamed 35 101 Dubaï Hiba Youssef 24 970 Casablanca Hiba Hafsa 22 131 Riyad Zahir Mohamed 35 654 Dubaï Saadi Fatima 39 555 Riyad Cherkaoui Hafsa 20 320 Riyad Neha Amina 29 145 Rabat Zahir Omar 35 120 Riyad Bouazza Ahmed 59 691 Le Caire Alami Mohamed 35 1030 Rabat Alami Fatima 33 321 Riyad Bouazza Leila 42 537 Fès Hamid Amina 34 944 Tanger Neha Khalid 58 824 Tunis Mansour Fatima 34 564 Marrakech Neha Khalid 29 496 Rabat Hamid Zahra 22 757 Riyad Oualid Mohamed 21 367 Alexandrie Bouazza Fatima 50 248 Marrakech Cherkaoui Youssef 53 550 Alexandrie Neha Youssef 55 431 Rabat Benjelloun Mohamed 25 521 Le Caire Hiba Khalid 41 224 Le Caire Mansour Fatima 20 621 Rabat Mansour Fatima 38 181 Marrakech Cherkaoui Ahmed 49 930 Casablanca Hiba Leila 34 252 Rabat Hiba Mohamed 41 891 Dubaï Idrissi Mohamed 42 828 Casablanca Bouazza Amina 34 998 Le Caire Mansour Mohamed 44 600 Alexandrie Idrissi Ahmed 24 414 Alexandrie Hamid Leila 38 958 Le Caire Hamid Ahmed 38 176 Dubaï Idrissi Mohamed 49 420 Fès Mansour Amina 53 486 Tanger Hamid Amina 40 842 Le Caire Zahir Khalid 54 1013 Riyad Qasimi Omar 49 790 Tunis Cherkaoui Ahmed 55 860 Tunis Saadi Leila 24 894 Tanger Qasimi Ahmed 40 399 Casablanca Alami Mohamed 43 267 Riyad El Fassi Ahmed 36 438 Fès Benjelloun Amina 24 214 Tanger Mansour Khalid 57 806 Tanger Qasimi Omar 53 333 Alexandrie Qasimi Fatima 30 745 Alexandrie El Fassi Mohamed 31 121 Le Caire Saadi Fatima 25 766 Tanger Qasimi Leila 32 846 Le Caire Bouazza Amina 28 115 Riyad Bouazza Mohamed 42 979 Marrakech Hamid Hafsa 35 546 Alexandrie Oualid Ahmed 37 895 Dubaï Saadi Youssef 47 263 Casablanca Idrissi Fatima 25 118 Fès Alami Youssef 37 536 Dubaï Hiba Zahra 57 575 Fès Zahir Ahmed 27 507 Alexandrie Zahir Zahra 30 217 Alexandrie Hamid Fatima 60 545 Tunis Bouazza Ahmed 45 133 Marrakech Mansour Omar 45 1043 Tunis Neha Amina 20 505 Fès Cherkaoui Omar 42 472 Rabat Hamid Zahra 36 271 Fès Benjelloun Mohamed 60 189 Alexandrie Oualid Omar 59 791 Alexandrie Alami Mohamed 44 904 Casablanca Bouazza Hafsa 25 785 Fès Mansour Khalid 35 996 Tanger Neha Fatima 44 289 Tanger Idrissi Hafsa 59 798 Tanger Bouazza Leila 28 1024 Fès Qasimi Ahmed 44 1052 Casablanca El Fassi Ahmed 35 132 Fès Alami Ahmed 52 816 Casablanca Bouazza Zahra 51 361 Tanger Saadi Ahmed 48 377 Rabat Alami Mohamed 27 407 Casablanca Saadi Ahmed 50 823 Rabat Idrissi Youssef 34 643 Casablanca Cherkaoui Youssef 41 344 Tanger Oualid Fatima 47 592 Rabat El Fassi Amina 33 831 Marrakech Oualid Ahmed 27 322 Marrakech Zahir Omar 60 735 Tanger Oualid Amina 27 937 Fès Alami Amina 30 663 Rabat Cherkaoui Omar 22 922 Le Caire Zahir Zahra 53 713 Le Caire Saadi Ahmed 29 668 Dubaï Bouazza Mohamed 32 731 Tanger Hamid Youssef 48 569 Tunis Mansour Leila 24 515 Rabat Cherkaoui Khalid 22 687 Dubaï Alami Mohamed 46 574 Riyad Cherkaoui Zahra 27 112 Marrakech Bouazza Mohamed 52 237 Tanger Hiba Khalid 45 139 Rabat Bouazza Omar 56 1049 Riyad Neha Amina 33 191 Marrakech Alami Hafsa 55 653 Dubaï Mansour Khalid 55 992 Dubaï Saadi Mohamed 39 617 Alexandrie Neha Hafsa 37 834 Alexandrie Bouazza Youssef 48 916 Rabat Saadi Hafsa 28 675 Tanger Neha Hafsa 59 803 Rabat Benjelloun Hafsa 57 209 Rabat Saadi Zahra 50 445 Riyad Qasimi Leila 56 758 Riyad Hamid Leila 60 100 Marrakech Hiba Ahmed 60 702 Marrakech Neha Khalid 31 142 Casablanca Zahir Amina 29 510 Tanger Neha Leila 41 727 Tanger Saadi Amina 52 457 Casablanca Benjelloun Hafsa 28 670 Tunis Benjelloun Mohamed 28 630 Marrakech Hiba Zahra 32 218 Le Caire Neha Zahra 25 1069 Rabat Idrissi Omar 50 512 Riyad Oualid Amina 32 925 Dubaï Hiba Fatima 48 146 Tanger Hiba Leila 27 287 Fès Benjelloun Ahmed 20 781 Casablanca Cherkaoui Khalid 43 974 Rabat Zahir Khalid 35 724 Tunis Alami Omar 46 625 Tanger Hamid Hafsa 40 200 Dubaï Bouazza Leila 48 273 Marrakech Idrissi Mohamed 24 869 Tunis Hamid Amina 24 801 Dubaï Hamid Hafsa 26 917 Dubaï Saadi Fatima 25 631 Marrakech Alami Leila 38 123 Tanger Benjelloun Khalid 57 1087 Dubaï Saadi Youssef 24 342 Casablanca Hiba Mohamed 20 942 Tanger Bouazza Khalid 52 720 Casablanca Neha Fatima 35 883 Riyad Hamid Omar 39 760 Tunis Zahir Omar 40 805 Dubaï Saadi Zahra 60 467 Tunis Oualid Leila 28 415 Alexandrie Benjelloun Hafsa 37 570 Tanger Alami Omar 20 1062 Tunis Cherkaoui Amina 31 396 Riyad Oualid Leila 20 230 Tanger Neha Leila 59 504 Riyad Qasimi Omar 31 162 Rabat Zahir Leila 45 542 Le Caire Alami Youssef 58 130 Le Caire Mansour Hafsa 45 577 Dubaï Oualid Ahmed 36 830 Dubaï Idrissi Zahra 28 516 Fès Mansour Mohamed 56 676 Rabat Qasimi Youssef 42 603 Le Caire Alami Amina 58 855 Riyad Cherkaoui Mohamed 60 755 Riyad Qasimi Amina 51 222 Tanger Zahir Omar 48 313 Riyad Alami Ahmed 54 911 Tanger Zahir Youssef 32 941 Fès Qasimi Youssef 44 955 Marrakech Idrissi Zahra 26 354 Alexandrie El Fassi Zahra 34 580 Tanger Hamid Hafsa 44 179 Casablanca Alami Omar 44 786 Le Caire El Fassi Mohamed 51 993 Rabat Neha Ahmed 29 1000 Alexandrie El Fassi Youssef 54 618 Fès Alami Amina 46 747 Marrakech Benjelloun Ahmed 59 780 Alexandrie Oualid Zahra 39 427 Tanger Zahir Omar 23 395 Rabat Oualid Mohamed 23 155 Tunis Oualid Ahmed 42 964 Marrakech Neha Ahmed 55 708 Tunis Hamid Zahra 29 330 Fès Mansour Mohamed 54 314 Casablanca Cherkaoui Youssef 47 318 Fès El Fassi Youssef 38 865 Le Caire Mansour Zahra 36 797 Dubaï Mansour Amina 29 915 Le Caire Benjelloun Ahmed 29 245 Marrakech Zahir Khalid 50 596 Marrakech El Fassi Zahra 30 173 Dubaï Oualid Fatima 41 1033 Tunis Idrissi Hafsa 46 409 Le Caire Alami Youssef 48 153 Marrakech Benjelloun Amina 27 116 Riyad Bouazza Leila 49 274 Tunis Zahir Omar 54 931 Marrakech Oualid Amina 37 1072 Fès Saadi Hafsa 24 442 Fès Benjelloun Khalid 32 919 Tunis Qasimi Khalid 49 556 Dubaï Neha Fatima 49 114 Rabat Qasimi Omar 28 349 Le Caire Qasimi Youssef 40 345 Rabat Saadi Fatima 27 400 Alexandrie Hamid Leila 60 954 Rabat Saadi Zahra 39 171 Rabat Qasimi Youssef 21 584 Dubaï Hiba Ahmed 36 939 Rabat Mansour Hafsa 35 532 Alexandrie Hiba Amina 60 560 Fès Saadi Khalid 38 166 Dubaï Saadi Hafsa 23 887 Riyad Hamid Youssef 56 723 Casablanca Mansour Fatima 36 934 Marrakech Idrissi Ahmed 40 335 Marrakech El Fassi Ahmed 44 302 Tanger Oualid Omar 60 109 Riyad Saadi Khalid 48 819 Tunis Hamid Youssef 24 770 Le Caire Cherkaoui Ahmed 46 826 Rabat Hamid Amina 48 658 Tunis Zahir Leila 41 362 Le Caire El Fassi Hafsa 35 1019 Tanger Idrissi Amina 33 439 Le Caire Benjelloun Leila 26 880 Fès Benjelloun Mohamed 48 1035 Le Caire Hamid Khalid 33 529 Marrakech Zahir Youssef 29 748 Alexandrie Zahir Youssef 22 642 Dubaï Bouazza Amina 41 1005 Marrakech Idrissi Amina 37 485 Fès Alami Youssef 52 283 Marrakech El Fassi Mohamed 41 315 Le Caire Mansour Youssef 23 614 Fès Benjelloun Amina 22 494 Tunis Saadi Ahmed 58 686 Le Caire Hamid Mohamed 29 432 Casablanca Zahir Fatima 36 151 Riyad Saadi Youssef 44 468 Marrakech Saadi Hafsa 43 655 Casablanca Neha Youssef 42 408 Fès Alami Ahmed 26 343 Riyad Oualid Mohamed 49 962 Casablanca Hamid Zahra 48 355 Casablanca Saadi Fatima 28 497 Alexandrie El Fassi Omar 37 893 Marrakech Hiba Ahmed 20 397 Alexandrie Qasimi Youssef 25 850 Alexandrie Alami Mohamed 24 890 Riyad Oualid Omar 47 125 Alexandrie Idrissi Youssef 30 402 Tanger Idrissi Ahmed 49 319 Riyad Benjelloun Omar 39 480 Alexandrie Benjelloun Hafsa 45 316 Fès Hamid Omar 20 606 Le Caire Zahir Youssef 48 1038 Alexandrie Saadi Mohamed 35 980 Riyad Qasimi Leila 57 360 Dubaï Saadi Hafsa 56 549 Fès Zahir Mohamed 21 364 Le Caire Zahir Zahra 40 462 Marrakech Mansour Mohamed 37 1016 Alexandrie Idrissi Mohamed 37 426 Rabat Hiba Mohamed 54 693 Tunis Cherkaoui Fatima 45 444 Dubaï Alami Amina 23 220 Fès Mansour Omar 27 807 Marrakech Bouazza Omar 60 629 Le Caire Hiba Zahra 26 1039 Marrakech Hamid Mohamed 50 352 Fès Hiba Youssef 47 1090 Fès Hamid Youssef 46 749 Casablanca Saadi Khalid 32 381 Marrakech Zahir Mohamed 44 1028 Tanger Cherkaoui Fatima 27 428 Le Caire Cherkaoui Fatima 45 483 Fès Idrissi Amina 25 524 Dubaï Bouazza Youssef 38 339 Fès et le convertie en json sous cette forme { "nom": "John", "prenom": "Doe", "age": 25, "numCompte": "1234567890", "ville": "Paris" }, { "nom": "Jane", "prenom": "Smith", "age": 32, "numCompte": "2345678901", "ville": "Lyon" } ]

**ChatGPT a dit :**

D'accord ! Voici un script Python qui lit les données à partir d'un fichier (ici le fichier fichierV11.txt), puis les convertit en une liste de dictionnaires au format JSON tel que demandé.

**Code Python :**

python

CopierModifier

import json *# Fonction pour transformer une ligne du fichier en dictionnaire* def convert\_to\_dict(line): *# On découpe la ligne en différents champs* parts = line.split() *# Si la ligne contient moins de 5 éléments, cela peut poser problème* if len(parts) < 5: return None *# Extraction des informations de la ligne* nom = parts[0] prenom = parts[1] age = int(parts[2]) *# Assurez-vous que l'âge est un nombre entier* num\_compte = parts[3] ville = " ".join(parts[4:]) *# On rejoint le reste de la ligne pour avoir la ville* *# Création d'un dictionnaire avec les données extraites* return { "nom": nom, "prenom": prenom, "age": age, "numCompte": num\_compte, "ville": ville } *# Fonction pour lire le fichier et convertir les lignes* def process\_file(file\_path): *# Liste pour stocker tous les enregistrements* data = [] *# Ouvrir et lire le fichier ligne par ligne* with open(file\_path, 'r') as file: for line in file: line = line.strip() *# On enlève les espaces inutiles aux bords de la ligne* record = convert\_to\_dict(line) if record: *# Si la ligne a été correctement convertie* data.append(record) return data *# Fonction principale pour créer le fichier JSON* def create\_json\_from\_file(file\_path, output\_path): *# Traiter le fichier pour obtenir les données sous forme de liste de dictionnaires* data = process\_file(file\_path) *# Enregistrer ces données dans un fichier JSON* with open(output\_path, 'w') as json\_file: json.dump(data, json\_file, indent=4, ensure\_ascii=False) print(f"Le fichier JSON a été créé avec succès à l'emplacement : {output\_path}") *# Spécifier le chemin d'entrée (votre fichier texte) et le chemin de sortie (fichier JSON)* input\_file = 'fichierV11.txt' *# Remplacez ce chemin par celui de votre fichier* output\_file = 'output.json' *# Le fichier JSON de sortie* *# Exécuter le script pour générer le fichier JSON* create\_json\_from\_file(input\_file, output\_file)

**Explication du script :**

1. **Fonction**convert\_to\_dict**:** Cette fonction prend une ligne du fichier, la découpe en morceaux (nom, prénom, âge, etc.), puis crée un dictionnaire avec ces données. Si la ligne ne contient pas assez de données pour être valide, elle renvoie None.
2. **Fonction**process\_file**:** Cette fonction ouvre le fichier ligne par ligne, utilise convert\_to\_dict pour convertir chaque ligne en un dictionnaire, et ajoute ces dictionnaires à une liste.
3. **Fonction**create\_json\_from\_file**:** Cette fonction appelle process\_file pour obtenir la liste des enregistrements, puis écrit cette liste dans un fichier JSON en utilisant json.dump.
4. **Fichier d'entrée :** Assurez-vous que le fichier texte (par exemple fichierV11.txt) est dans le bon format, et indiquez son chemin dans input\_file.
5. **Sortie :** Le script génère un fichier JSON nommé output.json contenant les données converties.

**Exemple de format JSON généré :**

json

CopierModifier

[ { "nom": "Neha", "prenom": "Amina", "age": 51, "numCompte": "1029", "ville": "Casablanca" }, { "nom": "Qasimi", "prenom": "Youssef", "age": 58, "numCompte": "548", "ville": "Tanger" }, { "nom": "Hamid", "prenom": "Hafsa", "age": 25, "numCompte": "370", "ville": "Fès" }, ... ]

**Instructions :**

1. **Remarque sur le fichier d'entrée :** Le fichier fichierV11.txt doit être présent dans le même répertoire que votre script Python (ou vous pouvez fournir un chemin absolu).
2. **Exécution du script :** Exécutez le script, et un fichier output.json sera créé dans le répertoire de votre projet.

Ce script répond à votre besoin de convertir un fichier texte avec des données structurées en un fichier JSON.

**Vous avez dit :**

je dois faire un projet en spring batch qui prend un fichier plat qui passe à travers un batch mapping et le converti en json step 1: chargement du fichier avec un file entete contenant id, nom, nb ligne, date et un fichier details dont lequel chaque ligne du fichier plat va etre dans une colonne (une ligne dans une case, et l'autre dans celle en dessous , etc...) puis on passe au processing ou on fait le mapping step 2: lecture configuration s'il te plait rend moi un code complet qui fait ce que j'ai demander qui prend le fichierV11 Neha Amina 51 1029 Casablanca Qasimi Youssef 58 548 Tanger Hamid Hafsa 25 370 Fès Neha Khalid 52 1095 Casablanca Benjelloun Fatima 58 452 Alexandrie Hiba Fatima 43 366 Marrakech Neha Omar 55 764 Le Caire Mansour Youssef 40 1097 Alexandrie Hamid Leila 49 721 Tunis Mansour Leila 23 952 Marrakech Bouazza Ahmed 35 223 Dubaï Zahir Hafsa 48 647 Marrakech Oualid Ahmed 26 710 Marrakech Cherkaoui Omar 36 499 Rabat Cherkaoui Youssef 43 573 Casablanca Hiba Amina 46 448 Marrakech Bouazza Leila 52 794 Fès Alami Omar 55 231 Le Caire Zahir Fatima 49 784 Rabat Cherkaoui Mohamed 42 707 Dubaï Benjelloun Zahra 30 726 Casablanca Cherkaoui Amina 32 382 Le Caire Idrissi Omar 35 429 Tunis Neha Youssef 20 744 Fès Neha Ahmed 59 697 Le Caire Bouazza Khalid 47 898 Casablanca Oualid Mohamed 56 619 Dubaï Bouazza Fatima 20 666 Dubaï Hiba Leila 24 789 Casablanca Hamid Omar 56 435 Tanger Hamid Mohamed 58 703 Casablanca Alami Khalid 34 193 Casablanca Zahir Hafsa 30 257 Dubaï Hamid Khalid 27 525 Casablanca Zahir Youssef 43 777 Tunis Idrissi Omar 44 698 Rabat Saadi Hafsa 26 501 Le Caire Zahir Khalid 24 795 Tunis Hamid Khalid 52 1023 Casablanca Oualid Omar 35 836 Le Caire Alami Youssef 53 157 Alexandrie Qasimi Mohamed 35 101 Dubaï Hiba Youssef 24 970 Casablanca Hiba Hafsa 22 131 Riyad Zahir Mohamed 35 654 Dubaï Saadi Fatima 39 555 Riyad Cherkaoui Hafsa 20 320 Riyad Neha Amina 29 145 Rabat Zahir Omar 35 120 Riyad Bouazza Ahmed 59 691 Le Caire Alami Mohamed 35 1030 Rabat Alami Fatima 33 321 Riyad Bouazza Leila 42 537 Fès Hamid Amina 34 944 Tanger Neha Khalid 58 824 Tunis Mansour Fatima 34 564 Marrakech Neha Khalid 29 496 Rabat Hamid Zahra 22 757 Riyad Oualid Mohamed 21 367 Alexandrie Bouazza Fatima 50 248 Marrakech Cherkaoui Youssef 53 550 Alexandrie Neha Youssef 55 431 Rabat Benjelloun Mohamed 25 521 Le Caire Hiba Khalid 41 224 Le Caire Mansour Fatima 20 621 Rabat Mansour Fatima 38 181 Marrakech Cherkaoui Ahmed 49 930 Casablanca Hiba Leila 34 252 Rabat Hiba Mohamed 41 891 Dubaï Idrissi Mohamed 42 828 Casablanca Bouazza Amina 34 998 Le Caire Mansour Mohamed 44 600 Alexandrie Idrissi Ahmed 24 414 Alexandrie Hamid Leila 38 958 Le Caire Hamid Ahmed 38 176 Dubaï Idrissi Mohamed 49 420 Fès Mansour Amina 53 486 Tanger Hamid Amina 40 842 Le Caire Zahir Khalid 54 1013 Riyad Qasimi Omar 49 790 Tunis Cherkaoui Ahmed 55 860 Tunis Saadi Leila 24 894 Tanger Qasimi Ahmed 40 399 Casablanca Alami Mohamed 43 267 Riyad El Fassi Ahmed 36 438 Fès Benjelloun Amina 24 214 Tanger Mansour Khalid 57 806 Tanger Qasimi Omar 53 333 Alexandrie Qasimi Fatima 30 745 Alexandrie El Fassi Mohamed 31 121 Le Caire Saadi Fatima 25 766 Tanger Qasimi Leila 32 846 Le Caire Bouazza Amina 28 115 Riyad Bouazza Mohamed 42 979 Marrakech Hamid Hafsa 35 546 Alexandrie Oualid Ahmed 37 895 Dubaï Saadi Youssef 47 263 Casablanca Idrissi Fatima 25 118 Fès Alami Youssef 37 536 Dubaï Hiba Zahra 57 575 Fès Zahir Ahmed 27 507 Alexandrie Zahir Zahra 30 217 Alexandrie Hamid Fatima 60 545 Tunis Bouazza Ahmed 45 133 Marrakech Mansour Omar 45 1043 Tunis Neha Amina 20 505 Fès Cherkaoui Omar 42 472 Rabat Hamid Zahra 36 271 Fès Benjelloun Mohamed 60 189 Alexandrie Oualid Omar 59 791 Alexandrie Alami Mohamed 44 904 Casablanca Bouazza Hafsa 25 785 Fès Mansour Khalid 35 996 Tanger Neha Fatima 44 289 Tanger Idrissi Hafsa 59 798 Tanger Bouazza Leila 28 1024 Fès Qasimi Ahmed 44 1052 Casablanca El Fassi Ahmed 35 132 Fès Alami Ahmed 52 816 Casablanca Bouazza Zahra 51 361 Tanger Saadi Ahmed 48 377 Rabat Alami Mohamed 27 407 Casablanca Saadi Ahmed 50 823 Rabat Idrissi Youssef 34 643 Casablanca Cherkaoui Youssef 41 344 Tanger Oualid Fatima 47 592 Rabat El Fassi Amina 33 831 Marrakech Oualid Ahmed 27 322 Marrakech Zahir Omar 60 735 Tanger Oualid Amina 27 937 Fès Alami Amina 30 663 Rabat Cherkaoui Omar 22 922 Le Caire Zahir Zahra 53 713 Le Caire Saadi Ahmed 29 668 Dubaï Bouazza Mohamed 32 731 Tanger Hamid Youssef 48 569 Tunis Mansour Leila 24 515 Rabat Cherkaoui Khalid 22 687 Dubaï Alami Mohamed 46 574 Riyad Cherkaoui Zahra 27 112 Marrakech Bouazza Mohamed 52 237 Tanger Hiba Khalid 45 139 Rabat Bouazza Omar 56 1049 Riyad Neha Amina 33 191 Marrakech Alami Hafsa 55 653 Dubaï Mansour Khalid 55 992 Dubaï Saadi Mohamed 39 617 Alexandrie Neha Hafsa 37 834 Alexandrie Bouazza Youssef 48 916 Rabat Saadi Hafsa 28 675 Tanger Neha Hafsa 59 803 Rabat Benjelloun Hafsa 57 209 Rabat Saadi Zahra 50 445 Riyad Qasimi Leila 56 758 Riyad Hamid Leila 60 100 Marrakech Hiba Ahmed 60 702 Marrakech Neha Khalid 31 142 Casablanca Zahir Amina 29 510 Tanger Neha Leila 41 727 Tanger Saadi Amina 52 457 Casablanca Benjelloun Hafsa 28 670 Tunis Benjelloun Mohamed 28 630 Marrakech Hiba Zahra 32 218 Le Caire Neha Zahra 25 1069 Rabat Idrissi Omar 50 512 Riyad Oualid Amina 32 925 Dubaï Hiba Fatima 48 146 Tanger Hiba Leila 27 287 Fès Benjelloun Ahmed 20 781 Casablanca Cherkaoui Khalid 43 974 Rabat Zahir Khalid 35 724 Tunis Alami Omar 46 625 Tanger Hamid Hafsa 40 200 Dubaï Bouazza Leila 48 273 Marrakech Idrissi Mohamed 24 869 Tunis Hamid Amina 24 801 Dubaï Hamid Hafsa 26 917 Dubaï Saadi Fatima 25 631 Marrakech Alami Leila 38 123 Tanger Benjelloun Khalid 57 1087 Dubaï Saadi Youssef 24 342 Casablanca Hiba Mohamed 20 942 Tanger Bouazza Khalid 52 720 Casablanca Neha Fatima 35 883 Riyad Hamid Omar 39 760 Tunis Zahir Omar 40 805 Dubaï Saadi Zahra 60 467 Tunis Oualid Leila 28 415 Alexandrie Benjelloun Hafsa 37 570 Tanger Alami Omar 20 1062 Tunis Cherkaoui Amina 31 396 Riyad Oualid Leila 20 230 Tanger Neha Leila 59 504 Riyad Qasimi Omar 31 162 Rabat Zahir Leila 45 542 Le Caire Alami Youssef 58 130 Le Caire Mansour Hafsa 45 577 Dubaï Oualid Ahmed 36 830 Dubaï Idrissi Zahra 28 516 Fès Mansour Mohamed 56 676 Rabat Qasimi Youssef 42 603 Le Caire Alami Amina 58 855 Riyad Cherkaoui Mohamed 60 755 Riyad Qasimi Amina 51 222 Tanger Zahir Omar 48 313 Riyad Alami Ahmed 54 911 Tanger Zahir Youssef 32 941 Fès Qasimi Youssef 44 955 Marrakech Idrissi Zahra 26 354 Alexandrie El Fassi Zahra 34 580 Tanger Hamid Hafsa 44 179 Casablanca Alami Omar 44 786 Le Caire El Fassi Mohamed 51 993 Rabat Neha Ahmed 29 1000 Alexandrie El Fassi Youssef 54 618 Fès Alami Amina 46 747 Marrakech Benjelloun Ahmed 59 780 Alexandrie Oualid Zahra 39 427 Tanger Zahir Omar 23 395 Rabat Oualid Mohamed 23 155 Tunis Oualid Ahmed 42 964 Marrakech Neha Ahmed 55 708 Tunis Hamid Zahra 29 330 Fès Mansour Mohamed 54 314 Casablanca Cherkaoui Youssef 47 318 Fès El Fassi Youssef 38 865 Le Caire Mansour Zahra 36 797 Dubaï Mansour Amina 29 915 Le Caire Benjelloun Ahmed 29 245 Marrakech Zahir Khalid 50 596 Marrakech El Fassi Zahra 30 173 Dubaï Oualid Fatima 41 1033 Tunis Idrissi Hafsa 46 409 Le Caire Alami Youssef 48 153 Marrakech Benjelloun Amina 27 116 Riyad Bouazza Leila 49 274 Tunis Zahir Omar 54 931 Marrakech Oualid Amina 37 1072 Fès Saadi Hafsa 24 442 Fès Benjelloun Khalid 32 919 Tunis Qasimi Khalid 49 556 Dubaï Neha Fatima 49 114 Rabat Qasimi Omar 28 349 Le Caire Qasimi Youssef 40 345 Rabat Saadi Fatima 27 400 Alexandrie Hamid Leila 60 954 Rabat Saadi Zahra 39 171 Rabat Qasimi Youssef 21 584 Dubaï Hiba Ahmed 36 939 Rabat Mansour Hafsa 35 532 Alexandrie Hiba Amina 60 560 Fès Saadi Khalid 38 166 Dubaï Saadi Hafsa 23 887 Riyad Hamid Youssef 56 723 Casablanca Mansour Fatima 36 934 Marrakech Idrissi Ahmed 40 335 Marrakech El Fassi Ahmed 44 302 Tanger Oualid Omar 60 109 Riyad Saadi Khalid 48 819 Tunis Hamid Youssef 24 770 Le Caire Cherkaoui Ahmed 46 826 Rabat Hamid Amina 48 658 Tunis Zahir Leila 41 362 Le Caire El Fassi Hafsa 35 1019 Tanger Idrissi Amina 33 439 Le Caire Benjelloun Leila 26 880 Fès Benjelloun Mohamed 48 1035 Le Caire Hamid Khalid 33 529 Marrakech Zahir Youssef 29 748 Alexandrie Zahir Youssef 22 642 Dubaï Bouazza Amina 41 1005 Marrakech Idrissi Amina 37 485 Fès Alami Youssef 52 283 Marrakech El Fassi Mohamed 41 315 Le Caire Mansour Youssef 23 614 Fès Benjelloun Amina 22 494 Tunis Saadi Ahmed 58 686 Le Caire Hamid Mohamed 29 432 Casablanca Zahir Fatima 36 151 Riyad Saadi Youssef 44 468 Marrakech Saadi Hafsa 43 655 Casablanca Neha Youssef 42 408 Fès Alami Ahmed 26 343 Riyad Oualid Mohamed 49 962 Casablanca Hamid Zahra 48 355 Casablanca Saadi Fatima 28 497 Alexandrie El Fassi Omar 37 893 Marrakech Hiba Ahmed 20 397 Alexandrie Qasimi Youssef 25 850 Alexandrie Alami Mohamed 24 890 Riyad Oualid Omar 47 125 Alexandrie Idrissi Youssef 30 402 Tanger Idrissi Ahmed 49 319 Riyad Benjelloun Omar 39 480 Alexandrie Benjelloun Hafsa 45 316 Fès Hamid Omar 20 606 Le Caire Zahir Youssef 48 1038 Alexandrie Saadi Mohamed 35 980 Riyad Qasimi Leila 57 360 Dubaï Saadi Hafsa 56 549 Fès Zahir Mohamed 21 364 Le Caire Zahir Zahra 40 462 Marrakech Mansour Mohamed 37 1016 Alexandrie Idrissi Mohamed 37 426 Rabat Hiba Mohamed 54 693 Tunis Cherkaoui Fatima 45 444 Dubaï Alami Amina 23 220 Fès Mansour Omar 27 807 Marrakech Bouazza Omar 60 629 Le Caire Hiba Zahra 26 1039 Marrakech Hamid Mohamed 50 352 Fès Hiba Youssef 47 1090 Fès Hamid Youssef 46 749 Casablanca Saadi Khalid 32 381 Marrakech Zahir Mohamed 44 1028 Tanger Cherkaoui Fatima 27 428 Le Caire Cherkaoui Fatima 45 483 Fès Idrissi Amina 25 524 Dubaï Bouazza Youssef 38 339 Fès et le convertie en json sous cette forme { "nom": "John", "prenom": "Doe", "age": 25, "numCompte": "1234567890", "ville": "Paris" }, { "nom": "Jane", "prenom": "Smith", "age": 32, "numCompte": "2345678901", "ville": "Lyon" } ]

**ChatGPT a dit :**

Pour réaliser ce projet Spring Batch en deux étapes comme vous l'avez décrit, voici un code complet qui charge un fichier plat, effectue le traitement nécessaire, puis le convertit en JSON.

Le projet Spring Batch comprendra deux étapes :

1. **Lecture du fichier plat et extraction des données.**
2. **Traitement des données et conversion en JSON.**

Voici les étapes détaillées et le code associé pour créer votre projet Spring Batch.

**1. Structure du projet**

Nous allons créer un projet Spring Boot avec Spring Batch pour exécuter le traitement.

**pom.xml (Dépendances nécessaires)**

xml

CopierModifier

<dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-json</artifactId> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-infrastructure</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-logging</artifactId> </dependency> </dependencies>

**2. Création des entités et des objets Java**

Nous allons créer une classe Person pour représenter chaque ligne de votre fichier.

java

CopierModifier

public class Person { private String firstName; private String lastName; private int age; private String numCompte; private String ville; *// Getters and Setters* }

**3. Création de la configuration Spring Batch**

Créez une configuration Spring Batch avec les étapes de traitement nécessaires.

**Batch Configuration**

java

CopierModifier

@Configuration @EnableBatchProcessing public class BatchConfig { @Autowired private JobBuilderFactory jobBuilderFactory; @Autowired private StepBuilderFactory stepBuilderFactory; @Autowired private ItemReader<Person> reader; @Autowired private ItemProcessor<Person, Person> processor; @Autowired private ItemWriter<Person> writer; @Bean public Job job() { return jobBuilderFactory.get("importPersonJob") .start(step1()) .build(); } @Bean public Step step1() { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) .reader(reader) .processor(processor) .writer(writer) .build(); } @Bean public FlatFileItemReader<Person> reader() { FlatFileItemReader<Person> reader = new FlatFileItemReader<>(); reader.setResource(new ClassPathResource("fichierV11.txt")); reader.setLineMapper(new DefaultLineMapper<Person>() {{ setLineTokenizer(new DelimitedLineTokenizer() {{ setNames("firstName", "lastName", "age", "numCompte", "ville"); }}); setFieldSetMapper(new BeanWrapperFieldSetMapper<>() {{ setTargetType(Person.class); }}); }}); return reader; } @Bean public ItemProcessor<Person, Person> processor() { return new ItemProcessor<Person, Person>() { @Override public Person process(Person person) throws Exception { *// Ajoutez tout traitement nécessaire ici* return person; } }; } @Bean public ItemWriter<Person> writer() { return new ItemWriter<Person>() { @Override public void write(List<? extends Person> items) throws Exception { ObjectMapper objectMapper = new ObjectMapper(); String json = objectMapper.writeValueAsString(items); System.out.println(json); *// Vous pouvez aussi l'écrire dans un fichier ou une base de données* } }; } }

**4. Exemple de fichier plat (**fichierV11.txt**)**

Votre fichier plat doit ressembler à ceci :

txt

CopierModifier

Neha,Amina,51,1029,Casablanca Qasimi,Youssef,58,548,Tanger Hamid,Hafsa,25,370,Fès Neha,Khalid,52,1095,Casablanca Benjelloun,Fatima,58,452,Alexandrie

**5. Démarrage du Job**

Créez un fichier principal pour démarrer votre application Spring Boot.

java

CopierModifier

@SpringBootApplication public class BatchApplication { public static void main(String[] args) { SpringApplication.run(BatchApplication.class, args); } }

**6. Structure JSON de Sortie**

Lorsque le batch s'exécute, il lit chaque ligne du fichier plat, applique le mapping avec le LineTokenizer, et les données sont transformées en une structure JSON similaire à celle-ci.

Sortie JSON attendue (exemple) :

json

CopierModifier

[ { "firstName": "Neha", "lastName": "Amina", "age": 51, "numCompte": "1029", "ville": "Casablanca" }, { "firstName": "Qasimi", "lastName": "Youssef", "age": 58, "numCompte": "548", "ville": "Tanger" }, { "firstName": "Hamid", "lastName": "Hafsa", "age": 25, "numCompte": "370", "ville": "Fès" }, ... ]

**7. Ajoutez un fichier de configuration pour le job**

Le fichier application.properties de Spring Boot pourrait ressembler à ceci :

properties

CopierModifier

spring.batch.job.enabled=false spring.batch.initialize-schema=always

**8. Conclusion**

Ce projet Spring Batch charge un fichier plat avec des données, applique un traitement (par exemple un mappage de données), et génère une sortie JSON pour chaque enregistrement. Vous pouvez ajouter des fonctionnalités supplémentaires selon vos besoins.

**Remarque**

Si vous souhaitez manipuler un plus grand nombre de lignes ou ajouter plus de logique de transformation, vous pouvez enrichir l'ItemProcessor pour effectuer des calculs ou autres transformations avant de convertir en JSON.

N'hésitez pas à ajuster le code selon vos besoins et à ajouter des éléments de personnalisation !

**Vous avez dit :**

je veux les codes complet sans que j'ajoute un point

**ChatGPT a dit :**

Voici un projet Spring Batch complet qui prend un fichier plat avec un en-tête, le traite dans un batch pour ensuite le convertir en JSON. Le code inclut les étapes de lecture, traitement, et écriture. Il est structuré de manière à respecter votre demande.

**Étapes du projet :**

1. **Lecture du fichier (file input)**.
2. **Transformation des données** (mappage).
3. **Écriture du fichier JSON**.

Je vais diviser le projet en plusieurs parties pour clarifier la structure.

**Structure des fichiers :**

plaintext

CopierModifier

- src - main - java - com - example - batch - config - BatchConfig.java - model - Person.java - processor - PersonItemProcessor.java - reader - FlatFileItemReader.java - writer - JsonFileItemWriter.java - resources - input - data.txt (fichier plat avec données) - application.properties

**1. Le modèle**Person**:**

java

CopierModifier

package com.example.batch.model; public class Person { private String nom; private String prenom; private int age; private String numCompte; private String ville; *// Getters et Setters* public String getNom() { return nom; } public void setNom(String nom) { this.nom = nom; } public String getPrenom() { return prenom; } public void setPrenom(String prenom) { this.prenom = prenom; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getNumCompte() { return numCompte; } public void setNumCompte(String numCompte) { this.numCompte = numCompte; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } }

**2. Le lecteur (**FlatFileItemReader**) :**

java

CopierModifier

package com.example.batch.reader; import com.example.batch.model.Person; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; import org.springframework.batch.item.file.separator.DefaultRecordSeparatorPolicy; import org.springframework.core.io.ClassPathResource; public class FlatFileItemReader extends FlatFileItemReader<Person> { public FlatFileItemReader() { setResource(new ClassPathResource("input/data.txt")); setLineMapper((line, lineNumber) -> { String[] fields = line.split("\\s+"); Person person = new Person(); person.setNom(fields[0]); person.setPrenom(fields[1]); person.setAge(Integer.parseInt(fields[2])); person.setNumCompte(fields[3]); person.setVille(fields[4]); return person; }); } }

**3. Le processeur (**PersonItemProcessor**) :**

java

CopierModifier

package com.example.batch.processor; import com.example.batch.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonItemProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person person) throws Exception { *// Transformation (si nécessaire)* return person; } }

**4. L'écrivain (**JsonFileItemWriter**) :**

java

CopierModifier

package com.example.batch.writer; import com.example.batch.model.Person; import org.springframework.batch.item.file.JsonFileItemWriter; import org.springframework.batch.core.StepExecution; import org.springframework.batch.core.scope.context.ChunkContext; import org.springframework.batch.core.scope.context.JobContext; import org.springframework.core.io.FileSystemResource; public class JsonFileItemWriter extends JsonFileItemWriter<Person> { public JsonFileItemWriter() { setResource(new FileSystemResource("output/data.json")); setJsonObjectMarshaller(new org.springframework.batch.item.file.transform.JsonMarshaller<>()); } @Override public void write(List<? extends Person> items) throws Exception { super.write(items); } }

**5. Configuration Spring Batch (**BatchConfig**) :**

java

CopierModifier

package com.example.batch.config; import com.example.batch.model.Person; import com.example.batch.processor.PersonItemProcessor; import com.example.batch.reader.FlatFileItemReader; import com.example.batch.writer.JsonFileItemWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.BatchConfigurer; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.step.builder.StepBuilderFactory; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.JsonFileItemWriter; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public FlatFileItemReader personItemReader() { return new FlatFileItemReader(); } @Bean public ItemProcessor<Person, Person> personItemProcessor() { return new PersonItemProcessor(); } @Bean public JsonFileItemWriter personItemWriter() { return new JsonFileItemWriter(); } @Bean public Step processStep(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("processStep") .<Person, Person>chunk(10) .reader(personItemReader()) .processor(personItemProcessor()) .writer(personItemWriter()) .build(); } @Bean public Job job(JobBuilderFactory jobBuilderFactory, Step processStep) { return jobBuilderFactory.get("processJob") .incrementer(new RunIdIncrementer()) .flow(processStep) .end() .build(); } }

**6. Fichier**application.properties**:**

properties

CopierModifier

spring.batch.job.enabled=false spring.batch.initialize-schema=never

**7. Fichier d'entrée**data.txt**(fichier plat) :**

Voici un exemple de format pour le fichier plat :

plaintext

CopierModifier

Neha Amina 51 1029 Casablanca Qasimi Youssef 58 548 Tanger Hamid Hafsa 25 370 Fès ...

**8. Exécution du projet :**

Le projet est configuré pour lire un fichier plat avec un format spécifique (nom, prénom, âge, numéro de compte, ville), effectuer un traitement (qui peut être étendu selon vos besoins) et enfin écrire le résultat en format JSON dans un fichier data.json.

**9. Exemple de fichier JSON produit :**

Le fichier JSON sera généré dans un format tel que celui-ci :

json

CopierModifier

[ { "nom": "Neha", "prenom": "Amina", "age": 51, "numCompte": "1029", "ville": "Casablanca" }, { "nom": "Qasimi", "prenom": "Youssef", "age": 58, "numCompte": "548", "ville": "Tanger" }, ... ]

**10. Lancer le Batch :**

Pour démarrer le batch, vous pouvez utiliser un simple main :

java

CopierModifier

package com.example.batch; import org.springframework.batch.core.launch.support.CommandLineJobRunner; import org.springframework.context.annotation.AnnotationConfigApplicationContext; public class BatchApplication { public static void main(String[] args) throws Exception { AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(); context.register(BatchConfig.class); context.refresh(); CommandLineJobRunner.main(new String[]{"classpath:applicationContext.xml", "processJob"}); } }

**Conclusion :**

Cela crée une solution complète pour traiter un fichier plat, mapper les données et les convertir en JSON avec Spring Batch. Vous n'avez pas besoin d'ajouter de points ou d'autres étapes supplémentaires. Il vous suffit de configurer correctement votre projet et d'exécuter le batch pour obtenir le fichier JSON.

**Vous avez dit :**

Traitement .idea target Traitement .mvn wrapper src main java Traitement config BatchConfig model Person processor PersonProcessor reader PersonReader writer TraitementApplication resources input fichierV11.txt output application.properties test target .gitattributes .gitignore HELP.md mvnw mvnw.cmd pom.xml

**ChatGPT a dit :**

D'accord ! Il semble que vous ayez partagé une structure de répertoire pour votre projet Spring Batch. Cette structure semble correspondre à un projet Java qui utilise Spring Batch et inclut des configurations et des composants spécifiques pour le traitement des données.

Je vais vous fournir une version améliorée et plus détaillée de ce que vous avez partagé, en corrigeant et en complétant le code pour que tout soit bien structuré dans votre projet. Je vais aussi adapter la structure de votre projet à la description fournie.

**Structure du Projet**

Voici la structure globale du projet avec des détails supplémentaires pour chaque partie.

plaintext

CopierModifier

Traitement ├── .idea # Configuration IntelliJ IDEA ├── target # Dossier généré par Maven après la compilation ├── .mvn # Répertoires liés à Maven ├── wrapper # Wrapper Maven ├── src │ ├── main │ │ ├── java │ │ │ └── Traitement │ │ │ ├── config │ │ │ │ └── BatchConfig.java # Configuration Spring Batch │ │ │ ├── model │ │ │ │ └── Person.java # Modèle de données │ │ │ ├── processor │ │ │ │ └── PersonProcessor.java # Logique de traitement des données │ │ │ ├── reader │ │ │ │ └── PersonReader.java # Lecteur des données du fichier plat │ │ │ ├── writer │ │ │ │ └── PersonWriter.java # Écrivain en format JSON │ │ │ └── TraitementApplication.java # Point d'entrée du projet │ │ └── resources │ │ ├── input │ │ │ └── fichierV11.txt # Fichier d'entrée avec les données │ │ ├── output │ │ └── application.properties # Configuration des propriétés de Spring Batch │ └── test │ └── java │ └── Traitement │ └── BatchTest.java # Tests unitaires pour Spring Batch ├── target ├── .gitignore # Fichiers ignorés par Git ├── .gitattributes ├── HELP.md ├── mvnw # Script pour exécuter Maven sous Linux/Mac ├── mvnw.cmd # Script pour exécuter Maven sous Windows └── pom.xml # Fichier de configuration de Maven

**1. Fichier**BatchConfig.java**(Configuration Spring Batch)**

java

CopierModifier

package Traitement.config; import Traitement.model.Person; import Traitement.processor.PersonProcessor; import Traitement.reader.PersonReader; import Traitement.writer.PersonWriter; import org.springframework.batch.core.Job; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.BatchConfigurer; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.step.builder.StepBuilderFactory; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.JsonFileItemWriter; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public FlatFileItemReader<Person> personItemReader() { return new PersonReader(); } @Bean public ItemProcessor<Person, Person> personItemProcessor() { return new PersonProcessor(); } @Bean public JsonFileItemWriter<Person> personItemWriter() { return new PersonWriter(); } @Bean public Step processStep(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("processStep") .<Person, Person>chunk(10) .reader(personItemReader()) .processor(personItemProcessor()) .writer(personItemWriter()) .build(); } @Bean public Job processJob(JobBuilderFactory jobBuilderFactory, Step processStep) { return jobBuilderFactory.get("processJob") .incrementer(new RunIdIncrementer()) .flow(processStep) .end() .build(); } }

**2. Modèle**Person.java

java

CopierModifier

package Traitement.model; public class Person { private String nom; private String prenom; private int age; private String numCompte; private String ville; *// Getters et Setters* public String getNom() { return nom; } public void setNom(String nom) { this.nom = nom; } public String getPrenom() { return prenom; } public void setPrenom(String prenom) { this.prenom = prenom; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getNumCompte() { return numCompte; } public void setNumCompte(String numCompte) { this.numCompte = numCompte; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } }

**3.**PersonProcessor.java**(Logique de transformation des données)**

java

CopierModifier

package Traitement.processor; import Traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person person) throws Exception { *// Appliquer la transformation ici si nécessaire (par exemple, modifier l'âge)* return person; } }

**4.**PersonReader.java**(Lecteur de données depuis le fichier plat)**

java

CopierModifier

package Traitement.reader; import Traitement.model.Person; import org.springframework.batch.item.file.FlatFileItemReader; import org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper; import org.springframework.batch.item.file.transform.DelimitedLineTokenizer; import org.springframework.core.io.ClassPathResource; public class PersonReader extends FlatFileItemReader<Person> { public PersonReader() { setResource(new ClassPathResource("input/fichierV11.txt")); setLineMapper((line, lineNumber) -> { String[] fields = line.split("\\s+"); Person person = new Person(); person.setNom(fields[0]); person.setPrenom(fields[1]); person.setAge(Integer.parseInt(fields[2])); person.setNumCompte(fields[3]); person.setVille(fields[4]); return person; }); } }

**5.**PersonWriter.java**(Écrivain JSON)**

java

CopierModifier

package Traitement.writer; import Traitement.model.Person; import org.springframework.batch.item.file.JsonFileItemWriter; import org.springframework.core.io.FileSystemResource; public class PersonWriter extends JsonFileItemWriter<Person> { public PersonWriter() { setResource(new FileSystemResource("output/data.json")); } }

**6. Application principale**TraitementApplication.java

java

CopierModifier

package Traitement; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.context.annotation.AnnotationConfigApplicationContext; import org.springframework.context.ApplicationContext; import org.springframework.batch.core.launch.support.CommandLineJobRunner; @EnableBatchProcessing public class TraitementApplication { public static void main(String[] args) throws Exception { ApplicationContext context = new AnnotationConfigApplicationContext(BatchConfig.class); JobLauncher jobLauncher = context.getBean(JobLauncher.class); Job job = context.getBean(Job.class); jobLauncher.run(job, new JobParameters()); } }

**7. Fichier d'entrée**fichierV11.txt**(Données)**

Exemple de données dans votre fichier input/fichierV11.txt :

plaintext

CopierModifier

Neha Amina 51 1029 Casablanca Qasimi Youssef 58 548 Tanger Hamid Hafsa 25 370 Fès ...

**8. Fichier de configuration**application.properties

properties

CopierModifier

spring.batch.job.enabled=false spring.batch.initialize-schema=never

**9. Fichier**pom.xml**(Maven)**

Voici un extrait du fichier pom.xml pour la configuration Maven de Spring Batch :

xml

CopierModifier

<dependencies> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>4.3.4</version> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifact

**Vous avez dit :**

package com.traitement.Traitement; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class TraitementApplication { public static void main(String[] args) { SpringApplication.run(TraitementApplication.class, args); } } package com.traitement.Traitement.writer; import com.traitement.Traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import org.springframework.batch.item.Chunk; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { this.objectMapper = new ObjectMapper(); } @Override public void write(Chunk<? extends Person> items) throws Exception { try { List<? extends Person> personList = items.getItems(); objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), personList); } catch (IOException e) { throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } } package com.traitement.Traitement.reader; import com.traitement.Traitement.model.Person; import org.springframework.batch.item.ItemReader; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; public class PersonReader implements ItemReader<Person> { private final String inputFile = "src/main/resources/input/fichierV11.txt"; private BufferedReader reader; private String currentLine; public PersonReader() { try { reader = new BufferedReader(new FileReader(inputFile)); } catch (IOException e) { e.printStackTrace(); } } @Override public Person read() throws Exception { currentLine = reader.readLine(); if (currentLine == null) { return null; } String[] values = currentLine.split(","); return new Person(values[0], values[1], Integer.parseInt(values[2]), values[3], values[4]); } public void close() throws IOException { if (reader != null) { reader.close(); } } } package com.traitement.Traitement.processor; import com.traitement.Traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) throws Exception { return item; } } package com.traitement.Traitement.model; public class Person { private String nom; private String prenoms; private int age; private String numcompte; private String ville; public Person(String nom, String prenoms, int age, String numcompte, String ville) { this.nom = nom; this.prenoms = prenoms; this.age = age; this.numcompte = numcompte; this.ville = ville; } public String getNom() { return nom; } public void setNom(String nom) { this.nom = nom; } public String getPrenoms() { return prenoms; } public void setPrenoms(String prenoms) { this.prenoms = prenoms; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getNumcompte() { return numcompte; } public void setNumcompte(String numcompte) { this.numcompte = numcompte; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } } <?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>3.4.4</version> <relativePath/> </parent> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>Traitement</name> <description>Demo project for Spring Boot with Batch processing</description> <properties> <java.version>17</java.version> </properties> <dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency> <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <optional>true</optional> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <configuration> <source>17</source> <target>17</target> <annotationProcessorPaths> <path> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </path> </annotationProcessorPaths> </configuration> </plugin> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project> Neha Amina 51 1029 Casablanca Qasimi Youssef 58 548 Tanger Hamid Hafsa 25 370 Fès Neha Khalid 52 1095 Casablanca Benjelloun Fatima 58 452 Alexandrie Hiba Fatima 43 366 Marrakech Neha Omar 55 764 Le Caire Mansour Youssef 40 1097 Alexandrie Hamid Leila 49 721 Tunis Mansour Leila 23 952 Marrakech Bouazza Ahmed 35 223 Dubaï Zahir Hafsa 48 647 Marrakech Oualid Ahmed 26 710 Marrakech Cherkaoui Omar 36 499 Rabat Cherkaoui Youssef 43 573 Casablanca Hiba Amina 46 448 Marrakech Bouazza Leila 52 794 Fès Alami Omar 55 231 Le Caire Zahir Fatima 49 784 Rabat Cherkaoui Mohamed 42 707 Dubaï Benjelloun Zahra 30 726 Casablanca Cherkaoui Amina 32 382 Le Caire Idrissi Omar 35 429 Tunis Neha Youssef 20 744 Fès Neha Ahmed 59 697 Le Caire Bouazza Khalid 47 898 Casablanca Oualid Mohamed 56 619 Dubaï Bouazza Fatima 20 666 Dubaï Hiba Leila 24 789 Casablanca Hamid Omar 56 435 Tanger Hamid Mohamed 58 703 Casablanca Alami Khalid 34 193 Casablanca Zahir Hafsa 30 257 Dubaï Hamid Khalid 27 525 Casablanca Zahir Youssef 43 777 Tunis Idrissi Omar 44 698 Rabat Saadi Hafsa 26 501 Le Caire Zahir Khalid 24 795 Tunis Hamid Khalid 52 1023 Casablanca Oualid Omar 35 836 Le Caire Alami Youssef 53 157 Alexandrie Qasimi Mohamed 35 101 Dubaï Hiba Youssef 24 970 Casablanca Hiba Hafsa 22 131 Riyad Zahir Mohamed 35 654 Dubaï Saadi Fatima 39 555 Riyad Cherkaoui Hafsa 20 320 Riyad Neha Amina 29 145 Rabat Zahir Omar 35 120 Riyad Bouazza Ahmed 59 691 Le Caire Alami Mohamed 35 1030 Rabat Alami Fatima 33 321 Riyad Bouazza Leila 42 537 Fès Hamid Amina 34 944 Tanger Neha Khalid 58 824 Tunis Mansour Fatima 34 564 Marrakech Neha Khalid 29 496 Rabat Hamid Zahra 22 757 Riyad Oualid Mohamed 21 367 Alexandrie Bouazza Fatima 50 248 Marrakech Cherkaoui Youssef 53 550 Alexandrie Neha Youssef 55 431 Rabat Benjelloun Mohamed 25 521 Le Caire Hiba Khalid 41 224 Le Caire Mansour Fatima 20 621 Rabat Mansour Fatima 38 181 Marrakech Cherkaoui Ahmed 49 930 Casablanca Hiba Leila 34 252 Rabat Hiba Mohamed 41 891 Dubaï Idrissi Mohamed 42 828 Casablanca Bouazza Amina 34 998 Le Caire Mansour Mohamed 44 600 Alexandrie Idrissi Ahmed 24 414 Alexandrie Hamid Leila 38 958 Le Caire Hamid Ahmed 38 176 Dubaï Idrissi Mohamed 49 420 Fès Mansour Amina 53 486 Tanger Hamid Amina 40 842 Le Caire Zahir Khalid 54 1013 Riyad Qasimi Omar 49 790 Tunis Cherkaoui Ahmed 55 860 Tunis Saadi Leila 24 894 Tanger Qasimi Ahmed 40 399 Casablanca Alami Mohamed 43 267 Riyad El Fassi Ahmed 36 438 Fès Benjelloun Amina 24 214 Tanger Mansour Khalid 57 806 Tanger Qasimi Omar 53 333 Alexandrie Qasimi Fatima 30 745 Alexandrie El Fassi Mohamed 31 121 Le Caire Saadi Fatima 25 766 Tanger Qasimi Leila 32 846 Le Caire Bouazza Amina 28 115 Riyad Bouazza Mohamed 42 979 Marrakech Hamid Hafsa 35 546 Alexandrie Oualid Ahmed 37 895 Dubaï Saadi Youssef 47 263 Casablanca Idrissi Fatima 25 118 Fès Alami Youssef 37 536 Dubaï Hiba Zahra 57 575 Fès Zahir Ahmed 27 507 Alexandrie Zahir Zahra 30 217 Alexandrie Hamid Fatima 60 545 Tunis Bouazza Ahmed 45 133 Marrakech Mansour Omar 45 1043 Tunis Neha Amina 20 505 Fès Cherkaoui Omar 42 472 Rabat Hamid Zahra 36 271 Fès Benjelloun Mohamed 60 189 Alexandrie Oualid Omar 59 791 Alexandrie Alami Mohamed 44 904 Casablanca Bouazza Hafsa 25 785 Fès Mansour Khalid 35 996 Tanger Neha Fatima 44 289 Tanger Idrissi Hafsa 59 798 Tanger Bouazza Leila 28 1024 Fès Qasimi Ahmed 44 1052 Casablanca El Fassi Ahmed 35 132 Fès Alami Ahmed 52 816 Casablanca Bouazza Zahra 51 361 Tanger Saadi Ahmed 48 377 Rabat Alami Mohamed 27 407 Casablanca Saadi Ahmed 50 823 Rabat Idrissi Youssef 34 643 Casablanca Cherkaoui Youssef 41 344 Tanger Oualid Fatima 47 592 Rabat El Fassi Amina 33 831 Marrakech Oualid Ahmed 27 322 Marrakech Zahir Omar 60 735 Tanger Oualid Amina 27 937 Fès Alami Amina 30 663 Rabat Cherkaoui Omar 22 922 Le Caire Zahir Zahra 53 713 Le Caire Saadi Ahmed 29 668 Dubaï Bouazza Mohamed 32 731 Tanger Hamid Youssef 48 569 Tunis Mansour Leila 24 515 Rabat Cherkaoui Khalid 22 687 Dubaï Alami Mohamed 46 574 Riyad Cherkaoui Zahra 27 112 Marrakech Bouazza Mohamed 52 237 Tanger Hiba Khalid 45 139 Rabat Bouazza Omar 56 1049 Riyad Neha Amina 33 191 Marrakech Alami Hafsa 55 653 Dubaï Mansour Khalid 55 992 Dubaï Saadi Mohamed 39 617 Alexandrie Neha Hafsa 37 834 Alexandrie Bouazza Youssef 48 916 Rabat Saadi Hafsa 28 675 Tanger Neha Hafsa 59 803 Rabat Benjelloun Hafsa 57 209 Rabat Saadi Zahra 50 445 Riyad Qasimi Leila 56 758 Riyad Hamid Leila 60 100 Marrakech Hiba Ahmed 60 702 Marrakech Neha Khalid 31 142 Casablanca Zahir Amina 29 510 Tanger Neha Leila 41 727 Tanger Saadi Amina 52 457 Casablanca Benjelloun Hafsa 28 670 Tunis Benjelloun Mohamed 28 630 Marrakech Hiba Zahra 32 218 Le Caire Neha Zahra 25 1069 Rabat Idrissi Omar 50 512 Riyad Oualid Amina 32 925 Dubaï Hiba Fatima 48 146 Tanger Hiba Leila 27 287 Fès Benjelloun Ahmed 20 781 Casablanca Cherkaoui Khalid 43 974 Rabat Zahir Khalid 35 724 Tunis Alami Omar 46 625 Tanger Hamid Hafsa 40 200 Dubaï Bouazza Leila 48 273 Marrakech Idrissi Mohamed 24 869 Tunis Hamid Amina 24 801 Dubaï Hamid Hafsa 26 917 Dubaï Saadi Fatima 25 631 Marrakech Alami Leila 38 123 Tanger Benjelloun Khalid 57 1087 Dubaï Saadi Youssef 24 342 Casablanca Hiba Mohamed 20 942 Tanger Bouazza Khalid 52 720 Casablanca Neha Fatima 35 883 Riyad Hamid Omar 39 760 Tunis Zahir Omar 40 805 Dubaï Saadi Zahra 60 467 Tunis Oualid Leila 28 415 Alexandrie Benjelloun Hafsa 37 570 Tanger Alami Omar 20 1062 Tunis Cherkaoui Amina 31 396 Riyad Oualid Leila 20 230 Tanger Neha Leila 59 504 Riyad Qasimi Omar 31 162 Rabat Zahir Leila 45 542 Le Caire Alami Youssef 58 130 Le Caire Mansour Hafsa 45 577 Dubaï Oualid Ahmed 36 830 Dubaï Idrissi Zahra 28 516 Fès Mansour Mohamed 56 676 Rabat Qasimi Youssef 42 603 Le Caire Alami Amina 58 855 Riyad Cherkaoui Mohamed 60 755 Riyad Qasimi Amina 51 222 Tanger Zahir Omar 48 313 Riyad Alami Ahmed 54 911 Tanger Zahir Youssef 32 941 Fès Qasimi Youssef 44 955 Marrakech Idrissi Zahra 26 354 Alexandrie El Fassi Zahra 34 580 Tanger Hamid Hafsa 44 179 Casablanca Alami Omar 44 786 Le Caire El Fassi Mohamed 51 993 Rabat Neha Ahmed 29 1000 Alexandrie El Fassi Youssef 54 618 Fès Alami Amina 46 747 Marrakech Benjelloun Ahmed 59 780 Alexandrie Oualid Zahra 39 427 Tanger Zahir Omar 23 395 Rabat Oualid Mohamed 23 155 Tunis Oualid Ahmed 42 964 Marrakech Neha Ahmed 55 708 Tunis Hamid Zahra 29 330 Fès Mansour Mohamed 54 314 Casablanca Cherkaoui Youssef 47 318 Fès El Fassi Youssef 38 865 Le Caire Mansour Zahra 36 797 Dubaï Mansour Amina 29 915 Le Caire Benjelloun Ahmed 29 245 Marrakech Zahir Khalid 50 596 Marrakech El Fassi Zahra 30 173 Dubaï Oualid Fatima 41 1033 Tunis Idrissi Hafsa 46 409 Le Caire Alami Youssef 48 153 Marrakech Benjelloun Amina 27 116 Riyad Bouazza Leila 49 274 Tunis Zahir Omar 54 931 Marrakech Oualid Amina 37 1072 Fès Saadi Hafsa 24 442 Fès Benjelloun Khalid 32 919 Tunis Qasimi Khalid 49 556 Dubaï Neha Fatima 49 114 Rabat Qasimi Omar 28 349 Le Caire Qasimi Youssef 40 345 Rabat Saadi Fatima 27 400 Alexandrie Hamid Leila 60 954 Rabat Saadi Zahra 39 171 Rabat Qasimi Youssef 21 584 Dubaï Hiba Ahmed 36 939 Rabat Mansour Hafsa 35 532 Alexandrie Hiba Amina 60 560 Fès Saadi Khalid 38 166 Dubaï Saadi Hafsa 23 887 Riyad Hamid Youssef 56 723 Casablanca Mansour Fatima 36 934 Marrakech Idrissi Ahmed 40 335 Marrakech El Fassi Ahmed 44 302 Tanger Oualid Omar 60 109 Riyad Saadi Khalid 48 819 Tunis Hamid Youssef 24 770 Le Caire Cherkaoui Ahmed 46 826 Rabat Hamid Amina 48 658 Tunis Zahir Leila 41 362 Le Caire El Fassi Hafsa 35 1019 Tanger Idrissi Amina 33 439 Le Caire Benjelloun Leila 26 880 Fès Benjelloun Mohamed 48 1035 Le Caire Hamid Khalid 33 529 Marrakech Zahir Youssef 29 748 Alexandrie Zahir Youssef 22 642 Dubaï Bouazza Amina 41 1005 Marrakech Idrissi Amina 37 485 Fès Alami Youssef 52 283 Marrakech El Fassi Mohamed 41 315 Le Caire Mansour Youssef 23 614 Fès Benjelloun Amina 22 494 Tunis Saadi Ahmed 58 686 Le Caire Hamid Mohamed 29 432 Casablanca Zahir Fatima 36 151 Riyad Saadi Youssef 44 468 Marrakech Saadi Hafsa 43 655 Casablanca Neha Youssef 42 408 Fès Alami Ahmed 26 343 Riyad Oualid Mohamed 49 962 Casablanca Hamid Zahra 48 355 Casablanca Saadi Fatima 28 497 Alexandrie El Fassi Omar 37 893 Marrakech Hiba Ahmed 20 397 Alexandrie Qasimi Youssef 25 850 Alexandrie Alami Mohamed 24 890 Riyad Oualid Omar 47 125 Alexandrie Idrissi Youssef 30 402 Tanger Idrissi Ahmed 49 319 Riyad Benjelloun Omar 39 480 Alexandrie Benjelloun Hafsa 45 316 Fès Hamid Omar 20 606 Le Caire Zahir Youssef 48 1038 Alexandrie Saadi Mohamed 35 980 Riyad Qasimi Leila 57 360 Dubaï Saadi Hafsa 56 549 Fès Zahir Mohamed 21 364 Le Caire Zahir Zahra 40 462 Marrakech Mansour Mohamed 37 1016 Alexandrie Idrissi Mohamed 37 426 Rabat Hiba Mohamed 54 693 Tunis Cherkaoui Fatima 45 444 Dubaï Alami Amina 23 220 Fès Mansour Omar 27 807 Marrakech Bouazza Omar 60 629 Le Caire Hiba Zahra 26 1039 Marrakech Hamid Mohamed 50 352 Fès Hiba Youssef 47 1090 Fès Hamid Youssef 46 749 Casablanca Saadi Khalid 32 381 Marrakech Zahir Mohamed 44 1028 Tanger Cherkaoui Fatima 27 428 Le Caire Cherkaoui Fatima 45 483 Fès Idrissi Amina 25 524 Dubaï Bouazza Youssef 38 339 Fès C:\Users\n.sahil\Downloads\jdk-17.0.12\_windows-x64\_bin\jdk-17.0.12\bin\java.exe -XX:TieredStopAtLevel=1 -Dspring.output.ansi.enabled=always -Dcom.sun.management.jmxremote -Dspring.jmx.enabled=true -Dspring.liveBeansView.mbeanDomain -Dspring.application.admin.enabled=true "-Dmanagement.endpoints.jmx.exposure.include=\*" "-javaagent:C:\Users\n.sahil\AppData\Local\JetBrains\IntelliJ IDEA 2025.1\lib\idea\_rt.jar=58656" -Dfile.encoding=UTF-8 -classpath C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-batch\3.4.4\spring-boot-starter-batch-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter\3.4.4\spring-boot-starter-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot\3.4.4\spring-boot-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-autoconfigure\3.4.4\spring-boot-autoconfigure-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-logging\3.4.4\spring-boot-starter-logging-3.4.4.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-classic\1.5.18\logback-classic-1.5.18.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-core\1.5.18\logback-core-1.5.18.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-to-slf4j\2.24.3\log4j-to-slf4j-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-api\2.24.3\log4j-api-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\jul-to-slf4j\2.0.17\jul-to-slf4j-2.0.17.jar;C:\Users\n.sahil\.m2\repository\jakarta\annotation\jakarta.annotation-api\2.1.1\jakarta.annotation-api-2.1.1.jar;C:\Users\n.sahil\.m2\repository\org\yaml\snakeyaml\2.3\snakeyaml-2.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-jdbc\3.4.4\spring-boot-starter-jdbc-3.4.4.jar;C:\Users\n.sahil\.m2\repository\com\zaxxer\HikariCP\5.1.0\HikariCP-5.1.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-data-jpa\3.4.4\spring-boot-starter-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\orm\hibernate-core\6.6.11.Final\hibernate-core-6.6.11.Final.jar;C:\Users\n.sahil\.m2\repository\jakarta\persistence\jakarta.persistence-api\3.1.0\jakarta.persistence-api-3.1.0.jar;C:\Users\n.sahil\.m2\repository\jakarta\transaction\jakarta.transaction-api\2.0.1\jakarta.transaction-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\jboss\logging\jboss-logging\3.6.1.Final\jboss-logging-3.6.1.Final.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\common\hibernate-commons-annotations\7.0.3.Final\hibernate-commons-annotations-7.0.3.Final.jar;C:\Users\n.sahil\.m2\repository\io\smallrye\jandex\3.2.0\jandex-3.2.0.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\classmate\1.7.0\classmate-1.7.0.jar;C:\Users\n.sahil\.m2\repository\net\bytebuddy\byte-buddy\1.15.11\byte-buddy-1.15.11.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-runtime\4.0.5\jaxb-runtime-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-core\4.0.5\jaxb-core-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\eclipse\angus\angus-activation\2.0.2\angus-activation-2.0.2.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\txw2\4.0.5\txw2-4.0.5.jar;C:\Users\n.sahil\.m2\repository\com\sun\istack\istack-commons-runtime\4.1.2\istack-commons-runtime-4.1.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\inject\jakarta.inject-api\2.0.1\jakarta.inject-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\antlr\antlr4-runtime\4.13.0\antlr4-runtime-4.13.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-jpa\3.4.4\spring-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-commons\3.4.4\spring-data-commons-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-orm\6.2.5\spring-orm-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\slf4j-api\2.0.17\slf4j-api-2.0.17.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aspects\6.2.5\spring-aspects-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\aspectj\aspectjweaver\1.9.23\aspectjweaver-1.9.23.jar;C:\Users\n.sahil\.m2\repository\org\postgresql\postgresql\42.7.5\postgresql-42.7.5.jar;C:\Users\n.sahil\.m2\repository\org\checkerframework\checker-qual\3.48.3\checker-qual-3.48.3.jar;C:\Users\n.sahil\.m2\repository\org\projectlombok\lombok\1.18.36\lombok-1.18.36.jar;C:\Users\n.sahil\.m2\repository\jakarta\xml\bind\jakarta.xml.bind-api\4.0.2\jakarta.xml.bind-api-4.0.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\activation\jakarta.activation-api\2.1.3\jakarta.activation-api-2.1.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-core\6.2.5\spring-core-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jcl\6.2.5\spring-jcl-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jdbc\6.2.5\spring-jdbc-6.2.5.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-databind\2.18.3\jackson-databind-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-annotations\2.18.3\jackson-annotations-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-core\2.18.3\jackson-core-2.18.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-core\5.0.0\spring-batch-core-5.0.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-infrastructure\5.2.2\spring-batch-infrastructure-5.2.2.jar;C:\Users\n.sahil\.m2\repository\org\springframework\retry\spring-retry\2.0.11\spring-retry-2.0.11.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aop\6.2.5\spring-aop-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-beans\6.2.5\spring-beans-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-context\6.2.5\spring-context-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-expression\6.2.5\spring-expression-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-tx\6.2.5\spring-tx-6.2.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-core\1.14.5\micrometer-core-1.14.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-commons\1.14.5\micrometer-commons-1.14.5.jar;C:\Users\n.sahil\.m2\repository\org\hdrhistogram\HdrHistogram\2.2.2\HdrHistogram-2.2.2.jar;C:\Users\n.sahil\.m2\repository\org\latencyutils\LatencyUtils\2.0.3\LatencyUtils-2.0.3.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-observation\1.14.5\micrometer-observation-1.14.5.jar com.traitement.Traitement.TraitementApplication . \_\_\_\_ \_ \_\_ \_ \_ /\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \ ( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_ | \ \ \ \ \\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) ) ' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / / =========|\_|==============|\_\_\_/=/\_/\_/\_/ :: Spring Boot :: (v3.4.4) 2025-04-23T15:53:19.561+01:00 INFO 9060 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : Starting TraitementApplication using Java 17.0.12 with PID 9060 (C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes started by n.sahil in C:\Users\n.sahil\Downloads\Traitement) 2025-04-23T15:53:19.566+01:00 INFO 9060 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : No active profile set, falling back to 1 default profile: "default" 2025-04-23T15:53:20.194+01:00 INFO 9060 --- [Traitement] [ main] o.s.b.c.c.annotation.BatchRegistrar : Finished Spring Batch infrastructure beans configuration in 11 ms. java.io.FileNotFoundException: src\main\resources\input\fichierV11.txt (Le chemin d’accès spécifié est introuvable) at java.base/java.io.FileInputStream.open0(Native Method) at java.base/java.io.FileInputStream.open(FileInputStream.java:216) at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157) at java.base/java.io.FileInputStream.<init>(FileInputStream.java:111) at java.base/java.io.FileReader.<init>(FileReader.java:60) at com.traitement.Traitement.reader.PersonReader.<init>(PersonReader.java:18) at com.traitement.Traitement.config.BatchConfig.reader(BatchConfig.java:24) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$0.CGLIB$reader$0(<generated>) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$FastClass$$1.invoke(<generated>) at org.springframework.cglib.proxy.MethodProxy.invokeSuper(MethodProxy.java:258) at org.springframework.context.annotation.ConfigurationClassEnhancer$BeanMethodInterceptor.intercept(ConfigurationClassEnhancer.java:370) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$0.reader(<generated>) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.base/java.lang.reflect.Method.invoke(Method.java:568) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.lambda$instantiate$0(SimpleInstantiationStrategy.java:171) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiateWithFactoryMethod(SimpleInstantiationStrategy.java:88) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiate(SimpleInstantiationStrategy.java:168) at org.springframework.beans.factory.support.ConstructorResolver.instantiate(ConstructorResolver.java:653) at org.springframework.beans.factory.support.ConstructorResolver.instantiateUsingFactoryMethod(ConstructorResolver.java:489) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.instantiateUsingFactoryMethod(AbstractAutowireCapableBeanFactory.java:1361) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBeanInstance(AbstractAutowireCapableBeanFactory.java:1191) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:563) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:523) at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:339) at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:347) at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:337) at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:202) at org.springframework.beans.factory.support.DefaultListableBeanFactory.instantiateSingleton(DefaultListableBeanFactory.java:1155) at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingleton(DefaultListableBeanFactory.java:1121) at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:1056) at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:987) at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:627) at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:752) at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:439) at org.springframework.boot.SpringApplication.run(SpringApplication.java:318) at org.springframework.boot.SpringApplication.run(SpringApplication.java:1361) at org.springframework.boot.SpringApplication.run(SpringApplication.java:1350) at com.traitement.Traitement.TraitementApplication.main(TraitementApplication.java:11) 2025-04-23T15:53:21.439+01:00 WARN 9060 --- [Traitement] [ main] s.c.a.AnnotationConfigApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'step1' defined in class path resource [com/traitement/Traitement/config/BatchConfig.class]: Unsatisfied dependency expressed through method 'step1' parameter 0: No qualifying bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' available: expected at least 1 bean which qualifies as autowire candidate. Dependency annotations: {} 2025-04-23T15:53:21.472+01:00 INFO 9060 --- [Traitement] [ main] .s.b.a.l.ConditionEvaluationReportLogger : Error starting ApplicationContext. To display the condition evaluation report re-run your application with 'debug' enabled. 2025-04-23T15:53:21.552+01:00 ERROR 9060 --- [Traitement] [ main] o.s.b.d.LoggingFailureAnalysisReporter : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* APPLICATION FAILED TO START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Description: Parameter 0 of method step1 in com.traitement.Traitement.config.BatchConfig required a bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' that could not be found. Action: Consider defining a bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' in your configuration. Process finished with exit code 1 ne change rien dans la structure du code juste courrige l'erreur sans modifier le code totalement

**ChatGPT a dit :**

​

Something went wrong, please try reloading the conversation.

**Vous avez dit :**

package com.traitement.Traitement; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class TraitementApplication { public static void main(String[] args) { SpringApplication.run(TraitementApplication.class, args); } } package com.itement.Traitement.writer; import com.traitement.Traitement.model.Person; import com.fasterxml.jackson.databind.ObjectMapper; import org.springframework.batch.item.ItemWriter; import org.springframework.batch.item.Chunk; import java.io.File; import java.io.IOException; import java.util.List; public class PersonWriter implements ItemWriter<Person> { private static final String OUTPUT\_FILE\_PATH = "src/main/resources/output/result.json"; private final ObjectMapper objectMapper; public PersonWriter() { this.objectMapper = new ObjectMapper(); } @Override public void write(Chunk<? extends Person> items) throws Exception { try { List<? extends Person> personList = items.getItems(); objectMapper.writeValue(new File(OUTPUT\_FILE\_PATH), personList); } catch (IOException e) { throw new RuntimeException("Erreur lors de l'écriture du fichier JSON", e); } } } package com.traitement.Traitement.reader; import com.traitement.Traitement.model.Person; import org.springframework.batch.item.ItemReader; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; public class PersonReader implements ItemReader<Person> { private final String inputFile = "src/main/resources/input/fichierV11.txt"; private BufferedReader reader; private String currentLine; public PersonReader() { try { reader = new BufferedReader(new FileReader(inputFile)); } catch (IOException e) { e.printStackTrace(); } } @Override public Person read() throws Exception { currentLine = reader.readLine(); if (currentLine == null) { return null; } String[] values = currentLine.split(","); return new Person(values[0], values[1], Integer.parseInt(values[2]), values[3], values[4]); } public void close() throws IOException { if (reader != null) { reader.close(); } } } package com.traitement.Traitement.processor; import com.traitement.Traitement.model.Person; import org.springframework.batch.item.ItemProcessor; public class PersonProcessor implements ItemProcessor<Person, Person> { @Override public Person process(Person item) throws Exception { return item; } } package com.traitement.Traitement.model; public class Person { private String nom; private String prenoms; private int age; private String numcompte; private String ville; public Person(String nom, String prenoms, int age, String numcompte, String ville) { this.nom = nom; this.prenoms = prenoms; this.age = age; this.numcompte = numcompte; this.ville = ville; } public String getNom() { return nom; } public void setNom(String nom) { this.nom = nom; } public String getPrenoms() { return prenoms; } public void setPrenoms(String prenoms) { this.prenoms = prenoms; } public int getAge() { return age; } public void setAge(int age) { this.age = age; } public String getNumcompte() { return numcompte; } public void setNumcompte(String numcompte) { this.numcompte = numcompte; } public String getVille() { return ville; } public void setVille(String ville) { this.ville = ville; } } <?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>3.4.4</version> <relativePath/> </parent> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>Traitement</name> <description>Demo project for Spring Boot with Batch processing</description> <properties> <java.version>17</java.version> </properties> <dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency> <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <optional>true</optional> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <configuration> <source>17</source> <target>17</target> <annotationProcessorPaths> <path> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </path> </annotationProcessorPaths> </configuration> </plugin> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project> Neha Amina 51 1029 Casablanca Qasimi Youssef 58 548 Tanger Hamid Hafsa 25 370 Fès Neha Khalid 52 1095 Casablanca Benjelloun Fatima 58 452 Alexandrie Hiba Fatima 43 366 Marrakech Neha Omar 55 764 Le Caire Mansour Youssef 40 1097 Alexandrie Hamid Leila 49 721 Tunis Mansour Leila 23 952 Marrakech Bouazza Ahmed 35 223 Dubaï Zahir Hafsa 48 647 Marrakech Oualid Ahmed 26 710 Marrakech Cherkaoui Omar 36 499 Rabat Cherkaoui Youssef 43 573 Casablanca Hiba Amina 46 448 Marrakech Bouazza Leila 52 794 Fès Alami Omar 55 231 Le Caire Zahir Fatima 49 784 Rabat Cherkaoui Mohamed 42 707 Dubaï Benjelloun Zahra 30 726 Casablanca Cherkaoui Amina 32 382 Le Caire Idrissi Omar 35 429 Tunis Neha Youssef 20 744 Fès Neha Ahmed 59 697 Le Caire Bouazza Khalid 47 898 Casablanca Oualid Mohamed 56 619 Dubaï Bouazza Fatima 20 666 Dubaï Hiba Leila 24 789 Casablanca Hamid Omar 56 435 Tanger Hamid Mohamed 58 703 Casablanca Alami Khalid 34 193 Casablanca Zahir Hafsa 30 257 Dubaï Hamid Khalid 27 525 Casablanca Zahir Youssef 43 777 Tunis Idrissi Omar 44 698 Rabat Saadi Hafsa 26 501 Le Caire Zahir Khalid 24 795 Tunis Hamid Khalid 52 1023 Casablanca Oualid Omar 35 836 Le Caire Alami Youssef 53 157 Alexandrie Qasimi Mohamed 35 101 Dubaï Hiba Youssef 24 970 Casablanca Hiba Hafsa 22 131 Riyad Zahir Mohamed 35 654 Dubaï Saadi Fatima 39 555 Riyad Cherkaoui Hafsa 20 320 Riyad Neha Amina 29 145 Rabat Zahir Omar 35 120 Riyad Bouazza Ahmed 59 691 Le Caire Alami Mohamed 35 1030 Rabat Alami Fatima 33 321 Riyad Bouazza Leila 42 537 Fès Hamid Amina 34 944 Tanger Neha Khalid 58 824 Tunis Mansour Fatima 34 564 Marrakech Neha Khalid 29 496 Rabat Hamid Zahra 22 757 Riyad Oualid Mohamed 21 367 Alexandrie Bouazza Fatima 50 248 Marrakech Cherkaoui Youssef 53 550 Alexandrie Neha Youssef 55 431 Rabat Benjelloun Mohamed 25 521 Le Caire Hiba Khalid 41 224 Le Caire Mansour Fatima 20 621 Rabat Mansour Fatima 38 181 Marrakech Cherkaoui Ahmed 49 930 Casablanca Hiba Leila 34 252 Rabat Hiba Mohamed 41 891 Dubaï Idrissi Mohamed 42 828 Casablanca Bouazza Amina 34 998 Le Caire Mansour Mohamed 44 600 Alexandrie Idrissi Ahmed 24 414 Alexandrie Hamid Leila 38 958 Le Caire Hamid Ahmed 38 176 Dubaï Idrissi Mohamed 49 420 Fès Mansour Amina 53 486 Tanger Hamid Amina 40 842 Le Caire Zahir Khalid 54 1013 Riyad Qasimi Omar 49 790 Tunis Cherkaoui Ahmed 55 860 Tunis Saadi Leila 24 894 Tanger Qasimi Ahmed 40 399 Casablanca Alami Mohamed 43 267 Riyad El Fassi Ahmed 36 438 Fès Benjelloun Amina 24 214 Tanger Mansour Khalid 57 806 Tanger Qasimi Omar 53 333 Alexandrie Qasimi Fatima 30 745 Alexandrie El Fassi Mohamed 31 121 Le Caire Saadi Fatima 25 766 Tanger Qasimi Leila 32 846 Le Caire Bouazza Amina 28 115 Riyad Bouazza Mohamed 42 979 Marrakech Hamid Hafsa 35 546 Alexandrie Oualid Ahmed 37 895 Dubaï Saadi Youssef 47 263 Casablanca Idrissi Fatima 25 118 Fès Alami Youssef 37 536 Dubaï Hiba Zahra 57 575 Fès Zahir Ahmed 27 507 Alexandrie Zahir Zahra 30 217 Alexandrie Hamid Fatima 60 545 Tunis Bouazza Ahmed 45 133 Marrakech Mansour Omar 45 1043 Tunis Neha Amina 20 505 Fès Cherkaoui Omar 42 472 Rabat Hamid Zahra 36 271 Fès Benjelloun Mohamed 60 189 Alexandrie Oualid Omar 59 791 Alexandrie Alami Mohamed 44 904 Casablanca Bouazza Hafsa 25 785 Fès Mansour Khalid 35 996 Tanger Neha Fatima 44 289 Tanger Idrissi Hafsa 59 798 Tanger Bouazza Leila 28 1024 Fès Qasimi Ahmed 44 1052 Casablanca El Fassi Ahmed 35 132 Fès Alami Ahmed 52 816 Casablanca Bouazza Zahra 51 361 Tanger Saadi Ahmed 48 377 Rabat Alami Mohamed 27 407 Casablanca Saadi Ahmed 50 823 Rabat Idrissi Youssef 34 643 Casablanca Cherkaoui Youssef 41 344 Tanger Oualid Fatima 47 592 Rabat El Fassi Amina 33 831 Marrakech Oualid Ahmed 27 322 Marrakech Zahir Omar 60 735 Tanger Oualid Amina 27 937 Fès Alami Amina 30 663 Rabat Cherkaoui Omar 22 922 Le Caire Zahir Zahra 53 713 Le Caire Saadi Ahmed 29 668 Dubaï Bouazza Mohamed 32 731 Tanger Hamid Youssef 48 569 Tunis Mansour Leila 24 515 Rabat Cherkaoui Khalid 22 687 Dubaï Alami Mohamed 46 574 Riyad Cherkaoui Zahra 27 112 Marrakech Bouazza Mohamed 52 237 Tanger Hiba Khalid 45 139 Rabat Bouazza Omar 56 1049 Riyad Neha Amina 33 191 Marrakech Alami Hafsa 55 653 Dubaï Mansour Khalid 55 992 Dubaï Saadi Mohamed 39 617 Alexandrie Neha Hafsa 37 834 Alexandrie Bouazza Youssef 48 916 Rabat Saadi Hafsa 28 675 Tanger Neha Hafsa 59 803 Rabat Benjelloun Hafsa 57 209 Rabat Saadi Zahra 50 445 Riyad Qasimi Leila 56 758 Riyad Hamid Leila 60 100 Marrakech Hiba Ahmed 60 702 Marrakech Neha Khalid 31 142 Casablanca Zahir Amina 29 510 Tanger Neha Leila 41 727 Tanger Saadi Amina 52 457 Casablanca Benjelloun Hafsa 28 670 Tunis Benjelloun Mohamed 28 630 Marrakech Hiba Zahra 32 218 Le Caire Neha Zahra 25 1069 Rabat Idrissi Omar 50 512 Riyad Oualid Amina 32 925 Dubaï Hiba Fatima 48 146 Tanger Hiba Leila 27 287 Fès Benjelloun Ahmed 20 781 Casablanca Cherkaoui Khalid 43 974 Rabat Zahir Khalid 35 724 Tunis Alami Omar 46 625 Tanger Hamid Hafsa 40 200 Dubaï Bouazza Leila 48 273 Marrakech Idrissi Mohamed 24 869 Tunis Hamid Amina 24 801 Dubaï Hamid Hafsa 26 917 Dubaï Saadi Fatima 25 631 Marrakech Alami Leila 38 123 Tanger Benjelloun Khalid 57 1087 Dubaï Saadi Youssef 24 342 Casablanca Hiba Mohamed 20 942 Tanger Bouazza Khalid 52 720 Casablanca Neha Fatima 35 883 Riyad Hamid Omar 39 760 Tunis Zahir Omar 40 805 Dubaï Saadi Zahra 60 467 Tunis Oualid Leila 28 415 Alexandrie Benjelloun Hafsa 37 570 Tanger Alami Omar 20 1062 Tunis Cherkaoui Amina 31 396 Riyad Oualid Leila 20 230 Tanger Neha Leila 59 504 Riyad Qasimi Omar 31 162 Rabat Zahir Leila 45 542 Le Caire Alami Youssef 58 130 Le Caire Mansour Hafsa 45 577 Dubaï Oualid Ahmed 36 830 Dubaï Idrissi Zahra 28 516 Fès Mansour Mohamed 56 676 Rabat Qasimi Youssef 42 603 Le Caire Alami Amina 58 855 Riyad Cherkaoui Mohamed 60 755 Riyad Qasimi Amina 51 222 Tanger Zahir Omar 48 313 Riyad Alami Ahmed 54 911 Tanger Zahir Youssef 32 941 Fès Qasimi Youssef 44 955 Marrakech Idrissi Zahra 26 354 Alexandrie El Fassi Zahra 34 580 Tanger Hamid Hafsa 44 179 Casablanca Alami Omar 44 786 Le Caire El Fassi Mohamed 51 993 Rabat Neha Ahmed 29 1000 Alexandrie El Fassi Youssef 54 618 Fès Alami Amina 46 747 Marrakech Benjelloun Ahmed 59 780 Alexandrie Oualid Zahra 39 427 Tanger Zahir Omar 23 395 Rabat Oualid Mohamed 23 155 Tunis Oualid Ahmed 42 964 Marrakech Neha Ahmed 55 708 Tunis Hamid Zahra 29 330 Fès Mansour Mohamed 54 314 Casablanca Cherkaoui Youssef 47 318 Fès El Fassi Youssef 38 865 Le Caire Mansour Zahra 36 797 Dubaï Mansour Amina 29 915 Le Caire Benjelloun Ahmed 29 245 Marrakech Zahir Khalid 50 596 Marrakech El Fassi Zahra 30 173 Dubaï Oualid Fatima 41 1033 Tunis Idrissi Hafsa 46 409 Le Caire Alami Youssef 48 153 Marrakech Benjelloun Amina 27 116 Riyad Bouazza Leila 49 274 Tunis Zahir Omar 54 931 Marrakech Oualid Amina 37 1072 Fès Saadi Hafsa 24 442 Fès Benjelloun Khalid 32 919 Tunis Qasimi Khalid 49 556 Dubaï Neha Fatima 49 114 Rabat Qasimi Omar 28 349 Le Caire Qasimi Youssef 40 345 Rabat Saadi Fatima 27 400 Alexandrie Hamid Leila 60 954 Rabat Saadi Zahra 39 171 Rabat Qasimi Youssef 21 584 Dubaï Hiba Ahmed 36 939 Rabat Mansour Hafsa 35 532 Alexandrie Hiba Amina 60 560 Fès Saadi Khalid 38 166 Dubaï Saadi Hafsa 23 887 Riyad Hamid Youssef 56 723 Casablanca Mansour Fatima 36 934 Marrakech Idrissi Ahmed 40 335 Marrakech El Fassi Ahmed 44 302 Tanger Oualid Omar 60 109 Riyad Saadi Khalid 48 819 Tunis Hamid Youssef 24 770 Le Caire Cherkaoui Ahmed 46 826 Rabat Hamid Amina 48 658 Tunis Zahir Leila 41 362 Le Caire El Fassi Hafsa 35 1019 Tanger Idrissi Amina 33 439 Le Caire Benjelloun Leila 26 880 Fès Benjelloun Mohamed 48 1035 Le Caire Hamid Khalid 33 529 Marrakech Zahir Youssef 29 748 Alexandrie Zahir Youssef 22 642 Dubaï Bouazza Amina 41 1005 Marrakech Idrissi Amina 37 485 Fès Alami Youssef 52 283 Marrakech El Fassi Mohamed 41 315 Le Caire Mansour Youssef 23 614 Fès Benjelloun Amina 22 494 Tunis Saadi Ahmed 58 686 Le Caire Hamid Mohamed 29 432 Casablanca Zahir Fatima 36 151 Riyad Saadi Youssef 44 468 Marrakech Saadi Hafsa 43 655 Casablanca Neha Youssef 42 408 Fès Alami Ahmed 26 343 Riyad Oualid Mohamed 49 962 Casablanca Hamid Zahra 48 355 Casablanca Saadi Fatima 28 497 Alexandrie El Fassi Omar 37 893 Marrakech Hiba Ahmed 20 397 Alexandrie Qasimi Youssef 25 850 Alexandrie Alami Mohamed 24 890 Riyad Oualid Omar 47 125 Alexandrie Idrissi Youssef 30 402 Tanger Idrissi Ahmed 49 319 Riyad Benjelloun Omar 39 480 Alexandrie Benjelloun Hafsa 45 316 Fès Hamid Omar 20 606 Le Caire Zahir Youssef 48 1038 Alexandrie Saadi Mohamed 35 980 Riyad Qasimi Leila 57 360 Dubaï Saadi Hafsa 56 549 Fès Zahir Mohamed 21 364 Le Caire Zahir Zahra 40 462 Marrakech Mansour Mohamed 37 1016 Alexandrie Idrissi Mohamed 37 426 Rabat Hiba Mohamed 54 693 Tunis Cherkaoui Fatima 45 444 Dubaï Alami Amina 23 220 Fès Mansour Omar 27 807 Marrakech Bouazza Omar 60 629 Le Caire Hiba Zahra 26 1039 Marrakech Hamid Mohamed 50 352 Fès Hiba Youssef 47 1090 Fès Hamid Youssef 46 749 Casablanca Saadi Khalid 32 381 Marrakech Zahir Mohamed 44 1028 Tanger Cherkaoui Fatima 27 428 Le Caire Cherkaoui Fatima 45 483 Fès Idrissi Amina 25 524 Dubaï Bouazza Youssef 38 339 Fès

**ChatGPT a dit :**

It looks like you've shared a Java Spring Batch project that processes a text file containing information about people (such as their names, ages, account numbers, and cities) and outputs the data in JSON format. The project includes several key components for the batch job: a reader, processor, writer, and a model class.

Here’s a breakdown of what each part of the code is doing:

1. TraitementApplication.java:
   * This is the main class that bootstraps the Spring Boot application with @SpringBootApplication.
2. PersonWriter.java:
   * This is an implementation of ItemWriter<Person> that writes a list of Person objects to a JSON file (result.json) using Jackson’s ObjectMapper.
3. PersonReader.java:
   * This is an implementation of ItemReader<Person> that reads a text file (fichierV11.txt), where each line represents a person’s data. It splits each line by commas and maps it to a Person object.
4. PersonProcessor.java:
   * This is an implementation of ItemProcessor<Person, Person> that processes each Person object. In your code, this processor currently doesn't alter the Person object (it just returns it as is).
5. Person.java:
   * A model class that represents a person with fields such as nom, prenoms, age, numcompte, and ville. It provides getter and setter methods for each field

**Vous avez dit :**

C:\Users\n.sahil\Downloads\jdk-17.0.12\_windows-x64\_bin\jdk-17.0.12\bin\java.exe -XX:TieredStopAtLevel=1 -Dspring.output.ansi.enabled=always -Dcom.sun.management.jmxremote -Dspring.jmx.enabled=true -Dspring.liveBeansView.mbeanDomain -Dspring.application.admin.enabled=true "-Dmanagement.endpoints.jmx.exposure.include=\*" "-javaagent:C:\Users\n.sahil\AppData\Local\JetBrains\IntelliJ IDEA 2025.1\lib\idea\_rt.jar=58656" -Dfile.encoding=UTF-8 -classpath C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-batch\3.4.4\spring-boot-starter-batch-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter\3.4.4\spring-boot-starter-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot\3.4.4\spring-boot-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-autoconfigure\3.4.4\spring-boot-autoconfigure-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-logging\3.4.4\spring-boot-starter-logging-3.4.4.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-classic\1.5.18\logback-classic-1.5.18.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-core\1.5.18\logback-core-1.5.18.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-to-slf4j\2.24.3\log4j-to-slf4j-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-api\2.24.3\log4j-api-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\jul-to-slf4j\2.0.17\jul-to-slf4j-2.0.17.jar;C:\Users\n.sahil\.m2\repository\jakarta\annotation\jakarta.annotation-api\2.1.1\jakarta.annotation-api-2.1.1.jar;C:\Users\n.sahil\.m2\repository\org\yaml\snakeyaml\2.3\snakeyaml-2.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-jdbc\3.4.4\spring-boot-starter-jdbc-3.4.4.jar;C:\Users\n.sahil\.m2\repository\com\zaxxer\HikariCP\5.1.0\HikariCP-5.1.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-data-jpa\3.4.4\spring-boot-starter-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\orm\hibernate-core\6.6.11.Final\hibernate-core-6.6.11.Final.jar;C:\Users\n.sahil\.m2\repository\jakarta\persistence\jakarta.persistence-api\3.1.0\jakarta.persistence-api-3.1.0.jar;C:\Users\n.sahil\.m2\repository\jakarta\transaction\jakarta.transaction-api\2.0.1\jakarta.transaction-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\jboss\logging\jboss-logging\3.6.1.Final\jboss-logging-3.6.1.Final.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\common\hibernate-commons-annotations\7.0.3.Final\hibernate-commons-annotations-7.0.3.Final.jar;C:\Users\n.sahil\.m2\repository\io\smallrye\jandex\3.2.0\jandex-3.2.0.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\classmate\1.7.0\classmate-1.7.0.jar;C:\Users\n.sahil\.m2\repository\net\bytebuddy\byte-buddy\1.15.11\byte-buddy-1.15.11.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-runtime\4.0.5\jaxb-runtime-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-core\4.0.5\jaxb-core-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\eclipse\angus\angus-activation\2.0.2\angus-activation-2.0.2.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\txw2\4.0.5\txw2-4.0.5.jar;C:\Users\n.sahil\.m2\repository\com\sun\istack\istack-commons-runtime\4.1.2\istack-commons-runtime-4.1.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\inject\jakarta.inject-api\2.0.1\jakarta.inject-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\antlr\antlr4-runtime\4.13.0\antlr4-runtime-4.13.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-jpa\3.4.4\spring-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-commons\3.4.4\spring-data-commons-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-orm\6.2.5\spring-orm-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\slf4j-api\2.0.17\slf4j-api-2.0.17.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aspects\6.2.5\spring-aspects-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\aspectj\aspectjweaver\1.9.23\aspectjweaver-1.9.23.jar;C:\Users\n.sahil\.m2\repository\org\postgresql\postgresql\42.7.5\postgresql-42.7.5.jar;C:\Users\n.sahil\.m2\repository\org\checkerframework\checker-qual\3.48.3\checker-qual-3.48.3.jar;C:\Users\n.sahil\.m2\repository\org\projectlombok\lombok\1.18.36\lombok-1.18.36.jar;C:\Users\n.sahil\.m2\repository\jakarta\xml\bind\jakarta.xml.bind-api\4.0.2\jakarta.xml.bind-api-4.0.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\activation\jakarta.activation-api\2.1.3\jakarta.activation-api-2.1.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-core\6.2.5\spring-core-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jcl\6.2.5\spring-jcl-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jdbc\6.2.5\spring-jdbc-6.2.5.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-databind\2.18.3\jackson-databind-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-annotations\2.18.3\jackson-annotations-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-core\2.18.3\jackson-core-2.18.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-core\5.0.0\spring-batch-core-5.0.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-infrastructure\5.2.2\spring-batch-infrastructure-5.2.2.jar;C:\Users\n.sahil\.m2\repository\org\springframework\retry\spring-retry\2.0.11\spring-retry-2.0.11.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aop\6.2.5\spring-aop-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-beans\6.2.5\spring-beans-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-context\6.2.5\spring-context-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-expression\6.2.5\spring-expression-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-tx\6.2.5\spring-tx-6.2.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-core\1.14.5\micrometer-core-1.14.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-commons\1.14.5\micrometer-commons-1.14.5.jar;C:\Users\n.sahil\.m2\repository\org\hdrhistogram\HdrHistogram\2.2.2\HdrHistogram-2.2.2.jar;C:\Users\n.sahil\.m2\repository\org\latencyutils\LatencyUtils\2.0.3\LatencyUtils-2.0.3.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-observation\1.14.5\micrometer-observation-1.14.5.jar com.traitement.Traitement.TraitementApplication . \_\_\_\_ \_ \_\_ \_ \_ /\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \ ( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_ | \ \ \ \ \\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) ) ' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / / =========|\_|==============|\_\_\_/=/\_/\_/\_/ :: Spring Boot :: (v3.4.4) 2025-04-23T15:53:19.561+01:00 INFO 9060 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : Starting TraitementApplication using Java 17.0.12 with PID 9060 (C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes started by n.sahil in C:\Users\n.sahil\Downloads\Traitement) 2025-04-23T15:53:19.566+01:00 INFO 9060 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : No active profile set, falling back to 1 default profile: "default" 2025-04-23T15:53:20.194+01:00 INFO 9060 --- [Traitement] [ main] o.s.b.c.c.annotation.BatchRegistrar : Finished Spring Batch infrastructure beans configuration in 11 ms. java.io.FileNotFoundException: src\main\resources\input\fichierV11.txt (Le chemin d’accès spécifié est introuvable) at java.base/java.io.FileInputStream.open0(Native Method) at java.base/java.io.FileInputStream.open(FileInputStream.java:216) at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157) at java.base/java.io.FileInputStream.<init>(FileInputStream.java:111) at java.base/java.io.FileReader.<init>(FileReader.java:60) at com.traitement.Traitement.reader.PersonReader.<init>(PersonReader.java:18) at com.traitement.Traitement.config.BatchConfig.reader(BatchConfig.java:24) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$0.CGLIB$reader$0(<generated>) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$FastClass$$1.invoke(<generated>) at org.springframework.cglib.proxy.MethodProxy.invokeSuper(MethodProxy.java:258) at org.springframework.context.annotation.ConfigurationClassEnhancer$BeanMethodInterceptor.intercept(ConfigurationClassEnhancer.java:370) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$0.reader(<generated>) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.base/java.lang.reflect.Method.invoke(Method.java:568) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.lambda$instantiate$0(SimpleInstantiationStrategy.java:171) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiateWithFactoryMethod(SimpleInstantiationStrategy.java:88) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiate(SimpleInstantiationStrategy.java:168) at org.springframework.beans.factory.support.ConstructorResolver.instantiate(ConstructorResolver.java:653) at org.springframework.beans.factory.support.ConstructorResolver.instantiateUsingFactoryMethod(ConstructorResolver.java:489) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.instantiateUsingFactoryMethod(AbstractAutowireCapableBeanFactory.java:1361) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBeanInstance(AbstractAutowireCapableBeanFactory.java:1191) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:563) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:523) at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:339) at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:347) at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:337) at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:202) at org.springframework.beans.factory.support.DefaultListableBeanFactory.instantiateSingleton(DefaultListableBeanFactory.java:1155) at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingleton(DefaultListableBeanFactory.java:1121) at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:1056) at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:987) at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:627) at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:752) at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:439) at org.springframework.boot.SpringApplication.run(SpringApplication.java:318) at org.springframework.boot.SpringApplication.run(SpringApplication.java:1361) at org.springframework.boot.SpringApplication.run(SpringApplication.java:1350) at com.traitement.Traitement.TraitementApplication.main(TraitementApplication.java:11) 2025-04-23T15:53:21.439+01:00 WARN 9060 --- [Traitement] [ main] s.c.a.AnnotationConfigApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'step1' defined in class path resource [com/traitement/Traitement/config/BatchConfig.class]: Unsatisfied dependency expressed through method 'step1' parameter 0: No qualifying bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' available: expected at least 1 bean which qualifies as autowire candidate. Dependency annotations: {} 2025-04-23T15:53:21.472+01:00 INFO 9060 --- [Traitement] [ main] .s.b.a.l.ConditionEvaluationReportLogger : Error starting ApplicationContext. To display the condition evaluation report re-run your application with 'debug' enabled. 2025-04-23T15:53:21.552+01:00 ERROR 9060 --- [Traitement] [ main] o.s.b.d.LoggingFailureAnalysisReporter : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* APPLICATION FAILED TO START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Description: Parameter 0 of method step1 in com.traitement.Traitement.config.BatchConfig required a bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' that could not be found. Action: Consider defining a bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' in your configuration. Process finished with exit code 1 ne change rien dans la structure du code juste courrige l'erreur sans modifier le code totalement

**ChatGPT a dit :**

L'erreur que tu rencontres indique qu'une dépendance requise, StepBuilderFactory, n'a pas été injectée correctement dans la configuration de ton batch Spring. Pour résoudre cette erreur sans modifier la structure du code, tu peux ajouter un @Bean pour StepBuilderFactory dans ta configuration Spring.

Voici ce que tu peux faire pour corriger cette erreur :

**Étape 1 : Ajouter un Bean pour**StepBuilderFactory

Dans ta classe BatchConfig, tu devras ajouter un @Bean pour définir un StepBuilderFactory. Cela permet de résoudre l'erreur liée au manque de ce bean.

java

CopierModifier

import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; @Configuration public class BatchConfig { private final JobBuilderFactory jobBuilderFactory; *// Injecter StepBuilderFactory* @Bean public StepBuilderFactory stepBuilderFactory() { return new StepBuilderFactory(jobRepository(), transactionManager()); *// Utilise le jobRepository et le transactionManager que tu as déjà dans ta configuration* } *// Ajoute d'autres configurations nécessaires ici...* }

**Étape 2 : Vérifier le JobRepository et TransactionManager**

L'StepBuilderFactory nécessite un JobRepository et un TransactionManager. Si tu les utilises déjà dans ta configuration, assure-toi qu'ils sont bien définis. Par exemple :

java

CopierModifier

@Bean public JobRepository jobRepository() { return new JobRepositoryBuilder() .dataSource(dataSource) .transactionManager(transactionManager()) .build(); } @Bean public PlatformTransactionManager transactionManager() { return new DataSourceTransactionManager(dataSource); }

**Résumé des modifications :**

* Ajouter un @Bean pour StepBuilderFactory dans ta configuration.
* Vérifier que les beans JobRepository et TransactionManager sont correctement définis.

Cela devrait corriger l'erreur sans nécessiter de modifications majeures dans ton code. Une fois que tu as ajouté ce bean, tu devrais pouvoir relancer ton application et éviter l'erreur liée à StepBuilderFactory.

**Vous avez dit :**

package com.traitement.Traitement.config; import com.traitement.Traitement.model.Person; import com.traitement.Traitement.processor.PersonProcessor; import com.traitement.Traitement.reader.PersonReader; import com.traitement.Traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemWriter; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public ItemReader<Person> reader() { return new PersonReader(); } @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); } @Bean public ItemWriter<Person> writer() { return new PersonWriter(); } @Bean public Step step1(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) .reader(reader()) .processor(processor()) .writer(writer()) .build(); } @Bean public Job processJob(JobBuilderFactory jobBuilderFactory, Step step1) { return jobBuilderFactory.get("processJob") .start(step1) .build(); } } donne moi un code complet qui ne génère pas d'erreur

**ChatGPT a dit :**

Pour résoudre l'erreur liée à l'absence de StepBuilderFactory, et en intégrant les éléments que tu as déjà définis dans ton code, voici la version complète de la configuration avec les correctifs nécessaires :

**Code complet de**BatchConfig.java

java

CopierModifier

package com.traitement.Traitement.config; import com.traitement.Traitement.model.Person; import com.traitement.Traitement.processor.PersonProcessor; import com.traitement.Traitement.reader.PersonReader; import com.traitement.Traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.configuration.annotation.DefaultBatchConfigurer; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemWriter; import org.springframework.batch.core.repository.JobRepository; import org.springframework.transaction.PlatformTransactionManager; import org.springframework.batch.core.explore.JobExplorer; import org.springframework.batch.core.launch.JobLauncher; @Configuration @EnableBatchProcessing public class BatchConfig extends DefaultBatchConfigurer { *// Injecter les beans nécessaires pour StepBuilderFactory* private final JobBuilderFactory jobBuilderFactory; private final StepBuilderFactory stepBuilderFactory; *// Constructor to inject the JobBuilderFactory and StepBuilderFactory* public BatchConfig(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory) { this.jobBuilderFactory = jobBuilderFactory; this.stepBuilderFactory = stepBuilderFactory; } *// Bean pour ItemReader* @Bean public ItemReader<Person> reader() { return new PersonReader(); } *// Bean pour ItemProcessor* @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); } *// Bean pour ItemWriter* @Bean public ItemWriter<Person> writer() { return new PersonWriter(); } *// Bean pour Step* @Bean public Step step1() { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) *// Taille du chunk* .reader(reader()) *// Définir le reader* .processor(processor()) *// Définir le processor* .writer(writer()) *// Définir le writer* .build(); } *// Bean pour Job* @Bean public Job processJob() { return jobBuilderFactory.get("processJob") .start(step1()) *// Définir le job en commençant par step1* .build(); } *// Les configurations de base pour la gestion des transactions et du JobRepository* @Override @Bean public JobRepository jobRepository() throws Exception { return super.jobRepository(); } @Override @Bean public PlatformTransactionManager transactionManager() throws Exception { return super.transactionManager(); } @Override @Bean public JobExplorer jobExplorer() throws Exception { return super.jobExplorer(); } @Override @Bean public JobLauncher jobLauncher() throws Exception { return super.jobLauncher(); } }

**Explication des modifications :**

1. **Injection de**JobBuilderFactory**et**StepBuilderFactory :
   * Tu injectes ces deux beans dans le constructeur de la classe BatchConfig. Ces beans sont nécessaires pour la configuration de ton job et de tes étapes.
2. **Beans pour le**JobRepository**et le**PlatformTransactionManager :
   * Si tu ne les as pas encore configurés ailleurs dans ton application, j'ai inclus ces méthodes pour te donner des configurations de base (ces beans sont nécessaires pour que Spring Batch fonctionne correctement).
3. @Bean**pour**Step**et**Job :
   * J'ai conservé la configuration que tu avais déjà pour définir step1 et processJob.
4. **Héritage de**DefaultBatchConfigurer :
   * J'ai fait hériter BatchConfig de DefaultBatchConfigurer, qui configure de manière automatique certains beans nécessaires à Spring Batch, comme JobRepository, PlatformTransactionManager, etc. Cela permet d'éviter de devoir les définir manuellement si tu utilises la configuration par défaut de Spring Batch.

**Autres points à vérifier :**

* **Chemin du fichier** : Assure-toi que le fichier fichierV11.txt existe bien dans le répertoire src/main/resources/input/ et qu'il est correctement accessible par ton application. Si ce n'est pas le cas, l'application échouera avec une erreur FileNotFoundException comme dans le log que tu as partagé précédemment.
* **Class**PersonReader**,**PersonProcessor**,**PersonWriter : Assure-toi que ces classes sont correctement définies, sans erreurs de compilation.

**Conclusion :**

Cette configuration devrait corriger l'erreur liée au StepBuilderFactory et permettre à ton batch de fonctionner sans erreurs supplémentaires. Tu peux maintenant tester et vérifier que ton application fonctionne correctement.

**Vous avez dit :**

C:\Users\n.sahil\Downloads\Traitement\Traitement\src\main\java\com\traitement\Traitement\config\BatchConfig.java:12:63 java: cannot find symbol symbol: class DefaultBatchConfigurer location: package org.springframework.batch.core.configuration.annotation

**ChatGPT a dit :**

L'erreur que tu rencontres est liée à l'absence de la classe DefaultBatchConfigurer. Il semble que tu n'aies pas la bonne version de la dépendance de Spring Batch dans ton projet, ou bien que cette classe ait été déplacée dans une autre version de Spring Batch.

Dans la dernière version de Spring Batch (Spring Batch 5.x), la classe DefaultBatchConfigurer a été déplacée ou modifiée.

**Solution :**

1. **Retirer**DefaultBatchConfigurer : Tu n'as pas besoin de cette classe dans Spring Batch 5.0 et versions ultérieures. Tu peux supprimer son utilisation dans ta classe BatchConfig et laisser Spring Boot gérer les beans de manière automatique.
2. **Simplification du code** : Je vais te fournir un exemple de configuration sans utiliser DefaultBatchConfigurer.

**Code corrigé sans**DefaultBatchConfigurer

java

CopierModifier

package com.traitement.Traitement.config; import com.traitement.Traitement.model.Person; import com.traitement.Traitement.processor.PersonProcessor; import com.traitement.Traitement.reader.PersonReader; import com.traitement.Traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemWriter; @Configuration @EnableBatchProcessing public class BatchConfig { private final JobBuilderFactory jobBuilderFactory; private final StepBuilderFactory stepBuilderFactory; *// Constructor to inject the JobBuilderFactory and StepBuilderFactory* public BatchConfig(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory) { this.jobBuilderFactory = jobBuilderFactory; this.stepBuilderFactory = stepBuilderFactory; } *// Bean pour ItemReader* @Bean public ItemReader<Person> reader() { return new PersonReader(); } *// Bean pour ItemProcessor* @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); } *// Bean pour ItemWriter* @Bean public ItemWriter<Person> writer() { return new PersonWriter(); } *// Bean pour Step* @Bean public Step step1() { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) *// Taille du chunk* .reader(reader()) *// Définir le reader* .processor(processor()) *// Définir le processor* .writer(writer()) *// Définir le writer* .build(); } *// Bean pour Job* @Bean public Job processJob() { return jobBuilderFactory.get("processJob") .start(step1()) *// Définir le job en commençant par step1* .build(); } }

**Explication des modifications :**

* **Suppression de**DefaultBatchConfigurer : Cette classe n'est plus nécessaire dans Spring Batch 5.x. Les beans nécessaires sont maintenant automatiquement gérés par Spring Boot grâce à l'annotation @EnableBatchProcessing que tu utilises déjà.
* **Pas de changements dans la configuration des beans** : J'ai conservé la définition des beans pour ItemReader, ItemProcessor, ItemWriter, Step et Job.

**Ce que tu dois faire ensuite :**

1. **Vérifie les dépendances de Spring Batch** : Assure-toi que ta version de spring-batch-core est bien compatible avec le code. Voici une dépendance de Spring Batch pour la version 5.x :

xml

CopierModifier

<dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> </dependency>

1. **Vérifie les autres dépendances de ton projet** : Assure-toi que toutes les dépendances sont correctement résolues dans ton pom.xml ou build.gradle (selon le système de gestion de dépendances que tu utilises).
2. **Test** : Essaie de lancer ton application à nouveau pour voir si l'erreur persiste.

Cela devrait résoudre ton problème sans nécessiter la classe DefaultBatchConfigurer et permettre à ton application Spring Batch de fonctionner correctement.

**Vous avez dit :**

C:\Users\n.sahil\Downloads\jdk-17.0.12\_windows-x64\_bin\jdk-17.0.12\bin\java.exe -XX:TieredStopAtLevel=1 -Dspring.output.ansi.enabled=always -Dcom.sun.management.jmxremote -Dspring.jmx.enabled=true -Dspring.liveBeansView.mbeanDomain -Dspring.application.admin.enabled=true "-Dmanagement.endpoints.jmx.exposure.include=\*" "-javaagent:C:\Users\n.sahil\AppData\Local\JetBrains\IntelliJ IDEA 2025.1\lib\idea\_rt.jar=58821" -Dfile.encoding=UTF-8 -classpath C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-batch\3.4.4\spring-boot-starter-batch-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter\3.4.4\spring-boot-starter-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot\3.4.4\spring-boot-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-autoconfigure\3.4.4\spring-boot-autoconfigure-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-logging\3.4.4\spring-boot-starter-logging-3.4.4.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-classic\1.5.18\logback-classic-1.5.18.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-core\1.5.18\logback-core-1.5.18.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-to-slf4j\2.24.3\log4j-to-slf4j-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-api\2.24.3\log4j-api-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\jul-to-slf4j\2.0.17\jul-to-slf4j-2.0.17.jar;C:\Users\n.sahil\.m2\repository\jakarta\annotation\jakarta.annotation-api\2.1.1\jakarta.annotation-api-2.1.1.jar;C:\Users\n.sahil\.m2\repository\org\yaml\snakeyaml\2.3\snakeyaml-2.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-jdbc\3.4.4\spring-boot-starter-jdbc-3.4.4.jar;C:\Users\n.sahil\.m2\repository\com\zaxxer\HikariCP\5.1.0\HikariCP-5.1.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-data-jpa\3.4.4\spring-boot-starter-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\orm\hibernate-core\6.6.11.Final\hibernate-core-6.6.11.Final.jar;C:\Users\n.sahil\.m2\repository\jakarta\persistence\jakarta.persistence-api\3.1.0\jakarta.persistence-api-3.1.0.jar;C:\Users\n.sahil\.m2\repository\jakarta\transaction\jakarta.transaction-api\2.0.1\jakarta.transaction-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\jboss\logging\jboss-logging\3.6.1.Final\jboss-logging-3.6.1.Final.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\common\hibernate-commons-annotations\7.0.3.Final\hibernate-commons-annotations-7.0.3.Final.jar;C:\Users\n.sahil\.m2\repository\io\smallrye\jandex\3.2.0\jandex-3.2.0.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\classmate\1.7.0\classmate-1.7.0.jar;C:\Users\n.sahil\.m2\repository\net\bytebuddy\byte-buddy\1.15.11\byte-buddy-1.15.11.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-runtime\4.0.5\jaxb-runtime-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-core\4.0.5\jaxb-core-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\eclipse\angus\angus-activation\2.0.2\angus-activation-2.0.2.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\txw2\4.0.5\txw2-4.0.5.jar;C:\Users\n.sahil\.m2\repository\com\sun\istack\istack-commons-runtime\4.1.2\istack-commons-runtime-4.1.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\inject\jakarta.inject-api\2.0.1\jakarta.inject-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\antlr\antlr4-runtime\4.13.0\antlr4-runtime-4.13.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-jpa\3.4.4\spring-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-commons\3.4.4\spring-data-commons-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-orm\6.2.5\spring-orm-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\slf4j-api\2.0.17\slf4j-api-2.0.17.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aspects\6.2.5\spring-aspects-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\aspectj\aspectjweaver\1.9.23\aspectjweaver-1.9.23.jar;C:\Users\n.sahil\.m2\repository\org\postgresql\postgresql\42.7.5\postgresql-42.7.5.jar;C:\Users\n.sahil\.m2\repository\org\checkerframework\checker-qual\3.48.3\checker-qual-3.48.3.jar;C:\Users\n.sahil\.m2\repository\org\projectlombok\lombok\1.18.36\lombok-1.18.36.jar;C:\Users\n.sahil\.m2\repository\jakarta\xml\bind\jakarta.xml.bind-api\4.0.2\jakarta.xml.bind-api-4.0.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\activation\jakarta.activation-api\2.1.3\jakarta.activation-api-2.1.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-core\6.2.5\spring-core-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jcl\6.2.5\spring-jcl-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jdbc\6.2.5\spring-jdbc-6.2.5.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-databind\2.18.3\jackson-databind-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-annotations\2.18.3\jackson-annotations-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-core\2.18.3\jackson-core-2.18.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-core\5.0.0\spring-batch-core-5.0.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-infrastructure\5.2.2\spring-batch-infrastructure-5.2.2.jar;C:\Users\n.sahil\.m2\repository\org\springframework\retry\spring-retry\2.0.11\spring-retry-2.0.11.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aop\6.2.5\spring-aop-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-beans\6.2.5\spring-beans-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-context\6.2.5\spring-context-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-expression\6.2.5\spring-expression-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-tx\6.2.5\spring-tx-6.2.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-core\1.14.5\micrometer-core-1.14.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-commons\1.14.5\micrometer-commons-1.14.5.jar;C:\Users\n.sahil\.m2\repository\org\hdrhistogram\HdrHistogram\2.2.2\HdrHistogram-2.2.2.jar;C:\Users\n.sahil\.m2\repository\org\latencyutils\LatencyUtils\2.0.3\LatencyUtils-2.0.3.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-observation\1.14.5\micrometer-observation-1.14.5.jar com.traitement.Traitement.TraitementApplication . \_\_\_\_ \_ \_\_ \_ \_ /\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \ ( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_ | \ \ \ \ \\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) ) ' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / / =========|\_|==============|\_\_\_/=/\_/\_/\_/ :: Spring Boot :: (v3.4.4) 2025-04-23T16:00:55.602+01:00 INFO 10244 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : Starting TraitementApplication using Java 17.0.12 with PID 10244 (C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes started by n.sahil in C:\Users\n.sahil\Downloads\Traitement) 2025-04-23T16:00:55.606+01:00 INFO 10244 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : No active profile set, falling back to 1 default profile: "default" 2025-04-23T16:00:56.210+01:00 INFO 10244 --- [Traitement] [ main] o.s.b.c.c.annotation.BatchRegistrar : Finished Spring Batch infrastructure beans configuration in 7 ms. 2025-04-23T16:00:56.807+01:00 WARN 10244 --- [Traitement] [ main] s.c.a.AnnotationConfigApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'batchConfig' defined in file [C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes\com\traitement\Traitement\config\BatchConfig.class]: Unsatisfied dependency expressed through constructor parameter 0: No qualifying bean of type 'org.springframework.batch.core.configuration.annotation.JobBuilderFactory' available: expected at least 1 bean which qualifies as autowire candidate. Dependency annotations: {} 2025-04-23T16:00:56.832+01:00 INFO 10244 --- [Traitement] [ main] .s.b.a.l.ConditionEvaluationReportLogger : Error starting ApplicationContext. To display the condition evaluation report re-run your application with 'debug' enabled. 2025-04-23T16:00:56.876+01:00 ERROR 10244 --- [Traitement] [ main] o.s.b.d.LoggingFailureAnalysisReporter : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* APPLICATION FAILED TO START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Description: Parameter 0 of constructor in com.traitement.Traitement.config.BatchConfig required a bean of type 'org.springframework.batch.core.configuration.annotation.JobBuilderFactory' that could not be found. Action: Consider defining a bean of type 'org.springframework.batch.core.configuration.annotation.JobBuilderFactory' in your configuration. Process finished with exit code 1

**ChatGPT a dit :**

L'erreur que tu rencontres indique que Spring Boot ne trouve pas de bean de type JobBuilderFactory lors de l'initialisation de ta configuration batch. Cela se produit généralement lorsqu'un composant nécessaire pour le traitement des jobs batch n'est pas correctement configuré dans ton contexte Spring.

**Problème :**

L'exception spécifique est :

vbnet

CopierModifier

UnsatisfiedDependencyException: Error creating bean with name *'batchConfig' defined in file [C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes\com\traitement\Traitement\config\BatchConfig.class]: Unsatisfied dependency expressed through constructor parameter 0: No qualifying bean of type 'org.springframework.batch.core.configuration.annotation.JobBuilderFactory' available: expected at least 1 bean which qualifies as autowire candidate.*

Cela signifie qu'il manque un bean JobBuilderFactory dans le contexte Spring.

**Solution :**

Tu dois t'assurer que Spring Batch est correctement configuré pour fournir les beans nécessaires. En particulier, JobBuilderFactory et StepBuilderFactory sont des composants essentiels pour construire les étapes et les jobs dans Spring Batch. Normalement, ces beans sont fournis automatiquement lorsque tu utilises l'annotation @EnableBatchProcessing.

**Modification de la classe de configuration**BatchConfig**:**

L'erreur est liée au fait que les JobBuilderFactory et StepBuilderFactory ne sont pas injectés automatiquement dans le constructeur de ta classe BatchConfig.

Voici comment tu peux corriger cela :

1. Tu n'as pas besoin de les injecter explicitement si tu utilises @EnableBatchProcessing. Spring Boot s'en charge automatiquement.
2. Tu peux utiliser une méthode qui ne passe pas par un constructeur pour les récupérer directement via la méthode jobBuilderFactory() et stepBuilderFactory() que Spring fournit.

**Code corrigé :**

java

CopierModifier

package com.traitement.Traitement.config; import com.traitement.Traitement.model.Person; import com.traitement.Traitement.processor.PersonProcessor; import com.traitement.Traitement.reader.PersonReader; import com.traitement.Traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemWriter; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public ItemReader<Person> reader() { return new PersonReader(); } @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); } @Bean public ItemWriter<Person> writer() { return new PersonWriter(); } *// Récupération de JobBuilderFactory et StepBuilderFactory via Spring* @Bean public Step step1(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) *// Taille du chunk* .reader(reader()) *// Définir le reader* .processor(processor()) *// Définir le processor* .writer(writer()) *// Définir le writer* .build(); } @Bean public Job processJob(JobBuilderFactory jobBuilderFactory, Step step1) { return jobBuilderFactory.get("processJob") .start(step1()) *// Définir le job en commençant par step1* .build(); } }

**Explications des changements :**

* **Utilisation de**JobBuilderFactory**et**StepBuilderFactory :
  + Ces deux beans sont automatiquement disponibles grâce à l'annotation @EnableBatchProcessing. Il n'est donc pas nécessaire de les injecter manuellement via le constructeur. Ils peuvent être injectés directement dans la méthode via des paramètres.
  + JobBuilderFactory et StepBuilderFactory sont automatiquement fournis par Spring Batch si tu as l'annotation @EnableBatchProcessing active.

**Vérification des dépendances :**

Vérifie que ton fichier pom.xml contient bien les bonnes dépendances Spring Batch, comme suit :

xml

CopierModifier

<dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> *<!-- Assure-toi que c'est une version compatible -->* </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> <version>3.4.4</version> *<!-- Vérifie la version de Spring Boot -->* </dependency>

**Redémarrage de l'application :**

Essaye de redémarrer l'application après avoir effectué ces modifications. Cela devrait résoudre le problème de dépendance non satisfaite pour JobBuilderFactory.

**Vous avez dit :**

C:\Users\n.sahil\Downloads\Traitement\Traitement\src\main\java\com\traitement\Traitement\config\BatchConfig.java:51:24 java: method step1 in class com.traitement.Traitement.config.BatchConfig cannot be applied to given types; required: org.springframework.batch.core.configuration.annotation.StepBuilderFactory found: no arguments reason: actual and formal argument lists differ in length

**ChatGPT a dit :**

L'erreur que tu rencontres indique que la méthode step1 attend un paramètre de type StepBuilderFactory, mais elle a été appelée sans argument dans la méthode processJob.

**Problème :**

Dans la méthode processJob, tu appelles la méthode step1() sans argument, mais step1 nécessite un paramètre de type StepBuilderFactory. Ce problème se produit parce que Spring Batch tente de résoudre le bean step1 en tant que méthode de retour mais n'a pas accès à StepBuilderFactory lorsqu'elle est appelée sans arguments.

**Solution :**

Pour résoudre ce problème, tu dois faire en sorte que step1 reçoive bien un paramètre de type StepBuilderFactory. Il est nécessaire de garder en tête que Spring gère l'injection des dépendances automatiquement, donc on n'a pas besoin de l'appeler explicitement sans les bonnes dépendances.

**Code modifié pour corriger l'erreur :**

Nous allons nous assurer que step1 reçoit correctement l'injection de StepBuilderFactory, en laissant Spring gérer l'appel de la méthode.

**Correction dans la méthode**processJob**:**

java

CopierModifier

package com.traitement.Traitement.config; import com.traitement.Traitement.model.Person; import com.traitement.Traitement.processor.PersonProcessor; import com.traitement.Traitement.reader.PersonReader; import com.traitement.Traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemWriter; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public ItemReader<Person> reader() { return new PersonReader(); } @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); } @Bean public ItemWriter<Person> writer() { return new PersonWriter(); } @Bean public Step step1(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) .reader(reader()) .processor(processor()) .writer(writer()) .build(); } @Bean public Job processJob(JobBuilderFactory jobBuilderFactory, Step step1) { return jobBuilderFactory.get("processJob") .start(step1) *// Utilisation de step1 sans parenthèses, car c'est un bean Spring* .build(); } }

**Explication des changements :**

1. **Injection de**StepBuilderFactory**dans**step1 :
   * La méthode step1(StepBuilderFactory stepBuilderFactory) reçoit un paramètre de type StepBuilderFactory, qui est automatiquement injecté par Spring grâce à l'annotation @EnableBatchProcessing.
2. **Correction de l'appel à**step1 :
   * Dans la méthode processJob, j'ai modifié l'appel à step1(). Il ne faut pas appeler step1() comme une méthode normale, mais simplement utiliser step1, car step1 est un bean Spring et Spring va l'injecter automatiquement.

**Vérification :**

Assure-toi que tu as bien les dépendances nécessaires dans ton pom.xml, surtout celles liées à Spring Batch, comme mentionné dans la solution précédente. Ensuite, relance ton application pour voir si l'erreur persiste.

Cela devrait résoudre le problème de la signature de la méthode et te permettre de continuer ton développement Batch.

**Vous avez dit :**

package com.traitement.Traitement.config; import com.traitement.Traitement.model.Person; import com.traitement.Traitement.processor.PersonProcessor; import com.traitement.Traitement.reader.PersonReader; import com.traitement.Traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.Job; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.batch.item.ItemReader; import org.springframework.batch.item.ItemProcessor; import org.springframework.batch.item.ItemWriter; @Configuration @EnableBatchProcessing public class BatchConfig { @Bean public ItemReader<Person> reader() { return new PersonReader(); } @Bean public ItemProcessor<Person, Person> processor() { return new PersonProcessor(); } @Bean public ItemWriter<Person> writer() { return new PersonWriter(); } @Bean public Step step1(StepBuilderFactory stepBuilderFactory) { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) .reader(reader()) .processor(processor()) .writer(writer()) .build(); } @Bean public Job processJob(JobBuilderFactory jobBuilderFactory, Step step1) { return jobBuilderFactory.get("processJob") .start(step1) // Utilisation de step1 sans parenthèses, car c'est un bean Spring .build(); } }

**ChatGPT a dit :**

Le code que tu as partagé semble correct pour une configuration de traitement par lots (batch) avec Spring Batch, à condition que toutes les dépendances nécessaires soient en place et correctement configurées dans ton projet. Voici un récapitulatif des points importants et des améliorations potentielles à vérifier si des erreurs subsistent :

**Points clés du code :**

1. @EnableBatchProcessing : Cette annotation active la configuration de Spring Batch, permettant de configurer un traitement par lots dans ton application.
2. **Définition des**ItemReader**,**ItemProcessor**, et**ItemWriter : Ces trois composants sont essentiels pour un traitement par lots. Ils respectent l'interface Spring Batch (ItemReader, ItemProcessor, ItemWriter).
3. StepBuilderFactory**et**JobBuilderFactory : Ces deux éléments sont nécessaires pour créer les Step et Job respectivement. Dans ton code, tu injectes ces dépendances via des méthodes @Bean.

**Étapes pour s'assurer que tout fonctionne correctement :**

1. **Vérifier les dépendances dans le**pom.xml : Assure-toi d'avoir bien ajouté les dépendances Spring Batch dans ton pom.xml. Voici un exemple de ce à quoi cela pourrait ressembler dans le fichier pom.xml :

xml

CopierModifier

<dependencies> *<!-- Spring Boot Starter Batch -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> *<!-- Spring Boot Starter Data JPA (si tu utilises JPA) -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> *<!-- Base Spring Boot Dependencies -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter</artifactId> </dependency> *<!-- Pour la base de données (si tu en as besoin) -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-jdbc</artifactId> </dependency> *<!-- Logging (optional) -->* <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-logging</artifactId> </dependency> *<!-- Lombok for convenience (optional) -->* <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <version>1.18.22</version> <scope>provided</scope> </dependency> </dependencies>

1. **Vérifier l'injection de dépendances** : Il est important que Spring Batch puisse injecter les beans nécessaires automatiquement. Vérifie que les méthodes step1 et processJob reçoivent bien les paramètres corrects (StepBuilderFactory et JobBuilderFactory). Cela devrait fonctionner automatiquement, étant donné que Spring s'en charge via le mécanisme d'injection de dépendances.
2. **Assurez-vous que les autres composants (**PersonReader**,**PersonProcessor**,**PersonWriter**) sont correctement définis** : Les classes comme PersonReader, PersonProcessor, et PersonWriter doivent implémenter correctement les interfaces de Spring Batch et ne pas contenir d'erreurs. Voici un rappel rapide de ce que chaque classe devrait faire :
   * PersonReader : Implémentation de l'interface ItemReader<Person>, qui lit les objets Person.
   * PersonProcessor : Implémentation de l'interface ItemProcessor<Person, Person>, qui effectue le traitement des objets Person si nécessaire.
   * PersonWriter : Implémentation de l'interface ItemWriter<Person>, qui écrit les objets Person dans la destination (base de données, fichier, etc.).

Assure-toi que ces classes sont correctement implémentées pour éviter toute erreur liée au traitement des données.

1. **Logs et erreurs de démarrage** : Si l'application échoue au démarrage, les logs de Spring Boot sont généralement détaillés. Par exemple, si une des dépendances requises est manquante ou mal configurée, Spring Boot te fournira un message d'erreur dans la console. Cela peut t'aider à diagnostiquer rapidement ce qui manque ou est mal configuré.
2. **Base de données ou autres sources de données** : Si tu utilises une base de données ou une autre source de données, vérifie bien que les configurations de connexion sont présentes dans ton fichier application.properties ou application.yml (si tu les utilises). Voici un exemple pour une base de données PostgreSQL dans application.properties :

properties

CopierModifier

spring.datasource.url=jdbc:postgresql://localhost:5432/ta\_base spring.datasource.username=ton\_username spring.datasource.password=ton\_mot\_de\_passe spring.datasource.driver-class-name=org.postgresql.Driver spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect spring.batch.initialize-schema=always

**Pour résumer :**

Ton code semble correct et devrait fonctionner pour une configuration de Spring Batch classique. Assure-toi que toutes les dépendances nécessaires sont présentes et que les autres composants sont correctement définis. Si des erreurs persistent, consulte les logs pour identifier les problèmes spécifiques liés aux beans ou aux dépendances manquantes.

**Vous avez dit :**

<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>3.4.4</version> <relativePath/> </parent> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>Traitement</name> <description>Demo project for Spring Boot with Batch processing</description> <properties> <java.version>17</java.version> </properties> <dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency> <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <optional>true</optional> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <configuration> <source>17</source> <target>17</target> <annotationProcessorPaths> <path> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </path> </annotationProcessorPaths> </configuration> </plugin> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project> voici mon pom dit ce quil faut ajouter

**ChatGPT a dit :**

Ton fichier pom.xml semble déjà assez complet, mais il manque quelques dépendances pour garantir que tout fonctionne correctement avec Spring Batch et Spring Boot. Je vais te détailler ce qu'il faut ajouter pour t'assurer que tout est bien configuré.

**Points à vérifier et potentiellement ajouter :**

1. **Dépendance pour Spring Batch** : Tu as déjà ajouté la dépendance spring-batch-core mais dans une version spécifique (5.0.0). Cependant, il est important que tu utilises une version qui est compatible avec Spring Boot 3.4.x.
   * La version de Spring Batch devrait être alignée avec ta version de Spring Boot, et donc, la version 5.0.0 pourrait être trop récente si tu utilises Spring Boot 3.4.4. **Spring Boot 3.x** est compatible avec **Spring Batch 5.x**, donc ta version 5.0.0 semble correcte.
2. **Spring Batch Infrastructure** : Bien que tu aies déjà spring-batch-core, si tu veux une configuration de batch complète, tu pourrais avoir besoin de la dépendance spring-batch-infrastructure pour accéder à toutes les fonctionnalités de Spring Batch, comme la gestion des job-executions et des step-executions. Elle est souvent incluse dans le spring-boot-starter-batch, mais si tu veux être plus explicite, tu pourrais ajouter ceci :

xml

CopierModifier

<dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-infrastructure</artifactId> <version>5.0.0</version> </dependency>

1. **Dépendance pour le JobRepository** : Pour Spring Batch, tu devras probablement configurer un JobRepository et un JobLauncher. Il faut donc s'assurer que Spring Batch crée correctement les tables nécessaires pour l'infrastructure de Batch dans la base de données.

En ajoutant la dépendance pour Spring Data JPA (que tu as déjà), tu devrais également vérifier que le schéma de la base de données de Spring Batch est bien créé automatiquement si tu utilises une base de données relationnelle (comme PostgreSQL dans ton cas).

Dans ton fichier application.properties, ajoute :

properties

CopierModifier

spring.batch.initialize-schema=always

Cela permet à Spring Batch de créer automatiquement les tables nécessaires au bon fonctionnement du traitement par lots. Si tu ne veux pas que Spring Batch crée des tables à chaque démarrage de l'application, tu peux définir spring.batch.initialize-schema=embedded pour ne créer les tables que si nécessaire.

1. **Version compatible de Spring Boot avec Spring Batch** : Si tu rencontres des problèmes de compatibilité, je te recommande de t'assurer que tu utilises des versions compatibles entre Spring Boot et Spring Batch. Voici les versions recommandées pour Spring Boot 3.4.x :
   * spring-boot-starter-batch: La version de Spring Boot que tu utilises (3.4.4) devrait être parfaitement compatible avec la version de Spring Batch 5.0.0 que tu as.
2. **Autres dépendances utiles (facultatives)** :
   * **Spring Data JPA** : Tu l'as déjà dans ton pom.xml, donc ça va bien pour l'intégration avec JPA.
   * **Logging** : Pour le logging de l'application, tu as spring-boot-starter-logging inclus dans spring-boot-starter, donc c'est parfait.

**Voici un exemple mis à jour de ton**pom.xml**:**

xml

CopierModifier

<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>3.4.4</version> <relativePath/> </parent> <groupId>com.traitement</groupId> <artifactId>Traitement</artifactId> <version>0.0.1-SNAPSHOT</version> <name>Traitement</name> <description>Demo project for Spring Boot with Batch processing</description> <properties> <java.version>17</java.version> </properties> <dependencies> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-batch</artifactId> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <scope>runtime</scope> </dependency> <dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <optional>true</optional> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-test</artifactId> <scope>test</scope> </dependency> <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> </dependency> <dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-infrastructure</artifactId> <version>5.0.0</version> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <configuration> <source>17</source> <target>17</target> <annotationProcessorPaths> <path> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </path> </annotationProcessorPaths> </configuration> </plugin> <plugin> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-maven-plugin</artifactId> </plugin> </plugins> </build> </project>

**Résumé :**

* La dépendance spring-batch-infrastructure est facultative, mais elle peut être ajoutée pour plus de clarté.
* Vérifie que tes entités et tes configurations de base de données sont correctes, en particulier les tables nécessaires pour Spring Batch (via spring.batch.initialize-schema dans application.properties).
* Si tu rencontres toujours des problèmes, assure-toi que les versions de Spring Boot et Spring Batch sont compatibles.

Cela devrait aider à résoudre la plupart des problèmes que tu pourrais rencontrer avec Spring Batch.

**Vous avez dit :**

C:\Users\n.sahil\Downloads\jdk-17.0.12\_windows-x64\_bin\jdk-17.0.12\bin\java.exe -XX:TieredStopAtLevel=1 -Dspring.output.ansi.enabled=always -Dcom.sun.management.jmxremote -Dspring.jmx.enabled=true -Dspring.liveBeansView.mbeanDomain -Dspring.application.admin.enabled=true "-Dmanagement.endpoints.jmx.exposure.include=\*" "-javaagent:C:\Users\n.sahil\AppData\Local\JetBrains\IntelliJ IDEA 2025.1\lib\idea\_rt.jar=58922" -Dfile.encoding=UTF-8 -classpath C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-batch\3.4.4\spring-boot-starter-batch-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter\3.4.4\spring-boot-starter-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot\3.4.4\spring-boot-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-autoconfigure\3.4.4\spring-boot-autoconfigure-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-logging\3.4.4\spring-boot-starter-logging-3.4.4.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-classic\1.5.18\logback-classic-1.5.18.jar;C:\Users\n.sahil\.m2\repository\ch\qos\logback\logback-core\1.5.18\logback-core-1.5.18.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-to-slf4j\2.24.3\log4j-to-slf4j-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\apache\logging\log4j\log4j-api\2.24.3\log4j-api-2.24.3.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\jul-to-slf4j\2.0.17\jul-to-slf4j-2.0.17.jar;C:\Users\n.sahil\.m2\repository\jakarta\annotation\jakarta.annotation-api\2.1.1\jakarta.annotation-api-2.1.1.jar;C:\Users\n.sahil\.m2\repository\org\yaml\snakeyaml\2.3\snakeyaml-2.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-jdbc\3.4.4\spring-boot-starter-jdbc-3.4.4.jar;C:\Users\n.sahil\.m2\repository\com\zaxxer\HikariCP\5.1.0\HikariCP-5.1.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\boot\spring-boot-starter-data-jpa\3.4.4\spring-boot-starter-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\orm\hibernate-core\6.6.11.Final\hibernate-core-6.6.11.Final.jar;C:\Users\n.sahil\.m2\repository\jakarta\persistence\jakarta.persistence-api\3.1.0\jakarta.persistence-api-3.1.0.jar;C:\Users\n.sahil\.m2\repository\jakarta\transaction\jakarta.transaction-api\2.0.1\jakarta.transaction-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\jboss\logging\jboss-logging\3.6.1.Final\jboss-logging-3.6.1.Final.jar;C:\Users\n.sahil\.m2\repository\org\hibernate\common\hibernate-commons-annotations\7.0.3.Final\hibernate-commons-annotations-7.0.3.Final.jar;C:\Users\n.sahil\.m2\repository\io\smallrye\jandex\3.2.0\jandex-3.2.0.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\classmate\1.7.0\classmate-1.7.0.jar;C:\Users\n.sahil\.m2\repository\net\bytebuddy\byte-buddy\1.15.11\byte-buddy-1.15.11.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-runtime\4.0.5\jaxb-runtime-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\jaxb-core\4.0.5\jaxb-core-4.0.5.jar;C:\Users\n.sahil\.m2\repository\org\eclipse\angus\angus-activation\2.0.2\angus-activation-2.0.2.jar;C:\Users\n.sahil\.m2\repository\org\glassfish\jaxb\txw2\4.0.5\txw2-4.0.5.jar;C:\Users\n.sahil\.m2\repository\com\sun\istack\istack-commons-runtime\4.1.2\istack-commons-runtime-4.1.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\inject\jakarta.inject-api\2.0.1\jakarta.inject-api-2.0.1.jar;C:\Users\n.sahil\.m2\repository\org\antlr\antlr4-runtime\4.13.0\antlr4-runtime-4.13.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-jpa\3.4.4\spring-data-jpa-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\data\spring-data-commons\3.4.4\spring-data-commons-3.4.4.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-orm\6.2.5\spring-orm-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\slf4j\slf4j-api\2.0.17\slf4j-api-2.0.17.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aspects\6.2.5\spring-aspects-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\aspectj\aspectjweaver\1.9.23\aspectjweaver-1.9.23.jar;C:\Users\n.sahil\.m2\repository\org\postgresql\postgresql\42.7.5\postgresql-42.7.5.jar;C:\Users\n.sahil\.m2\repository\org\checkerframework\checker-qual\3.48.3\checker-qual-3.48.3.jar;C:\Users\n.sahil\.m2\repository\org\projectlombok\lombok\1.18.36\lombok-1.18.36.jar;C:\Users\n.sahil\.m2\repository\jakarta\xml\bind\jakarta.xml.bind-api\4.0.2\jakarta.xml.bind-api-4.0.2.jar;C:\Users\n.sahil\.m2\repository\jakarta\activation\jakarta.activation-api\2.1.3\jakarta.activation-api-2.1.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-core\6.2.5\spring-core-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jcl\6.2.5\spring-jcl-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-jdbc\6.2.5\spring-jdbc-6.2.5.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-databind\2.18.3\jackson-databind-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-annotations\2.18.3\jackson-annotations-2.18.3.jar;C:\Users\n.sahil\.m2\repository\com\fasterxml\jackson\core\jackson-core\2.18.3\jackson-core-2.18.3.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-core\5.0.0\spring-batch-core-5.0.0.jar;C:\Users\n.sahil\.m2\repository\org\springframework\batch\spring-batch-infrastructure\5.2.2\spring-batch-infrastructure-5.2.2.jar;C:\Users\n.sahil\.m2\repository\org\springframework\retry\spring-retry\2.0.11\spring-retry-2.0.11.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-aop\6.2.5\spring-aop-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-beans\6.2.5\spring-beans-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-context\6.2.5\spring-context-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-expression\6.2.5\spring-expression-6.2.5.jar;C:\Users\n.sahil\.m2\repository\org\springframework\spring-tx\6.2.5\spring-tx-6.2.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-core\1.14.5\micrometer-core-1.14.5.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-commons\1.14.5\micrometer-commons-1.14.5.jar;C:\Users\n.sahil\.m2\repository\org\hdrhistogram\HdrHistogram\2.2.2\HdrHistogram-2.2.2.jar;C:\Users\n.sahil\.m2\repository\org\latencyutils\LatencyUtils\2.0.3\LatencyUtils-2.0.3.jar;C:\Users\n.sahil\.m2\repository\io\micrometer\micrometer-observation\1.14.5\micrometer-observation-1.14.5.jar com.traitement.Traitement.TraitementApplication . \_\_\_\_ \_ \_\_ \_ \_ /\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \ ( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_ | \ \ \ \ \\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) ) ' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / / =========|\_|==============|\_\_\_/=/\_/\_/\_/ :: Spring Boot :: (v3.4.4) 2025-04-23T16:05:59.988+01:00 INFO 19764 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : Starting TraitementApplication using Java 17.0.12 with PID 19764 (C:\Users\n.sahil\Downloads\Traitement\Traitement\target\classes started by n.sahil in C:\Users\n.sahil\Downloads\Traitement) 2025-04-23T16:05:59.995+01:00 INFO 19764 --- [Traitement] [ main] c.t.Traitement.TraitementApplication : No active profile set, falling back to 1 default profile: "default" 2025-04-23T16:06:00.741+01:00 INFO 19764 --- [Traitement] [ main] o.s.b.c.c.annotation.BatchRegistrar : Finished Spring Batch infrastructure beans configuration in 18 ms. java.io.FileNotFoundException: src\main\resources\input\fichierV11.txt (Le chemin d’accès spécifié est introuvable) at java.base/java.io.FileInputStream.open0(Native Method) at java.base/java.io.FileInputStream.open(FileInputStream.java:216) at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157) at java.base/java.io.FileInputStream.<init>(FileInputStream.java:111) at java.base/java.io.FileReader.<init>(FileReader.java:60) at com.traitement.Traitement.reader.PersonReader.<init>(PersonReader.java:18) at com.traitement.Traitement.config.BatchConfig.reader(BatchConfig.java:24) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$0.CGLIB$reader$0(<generated>) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$FastClass$$1.invoke(<generated>) at org.springframework.cglib.proxy.MethodProxy.invokeSuper(MethodProxy.java:258) at org.springframework.context.annotation.ConfigurationClassEnhancer$BeanMethodInterceptor.intercept(ConfigurationClassEnhancer.java:370) at com.traitement.Traitement.config.BatchConfig$$SpringCGLIB$$0.reader(<generated>) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.base/java.lang.reflect.Method.invoke(Method.java:568) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.lambda$instantiate$0(SimpleInstantiationStrategy.java:171) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiateWithFactoryMethod(SimpleInstantiationStrategy.java:88) at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiate(SimpleInstantiationStrategy.java:168) at org.springframework.beans.factory.support.ConstructorResolver.instantiate(ConstructorResolver.java:653) at org.springframework.beans.factory.support.ConstructorResolver.instantiateUsingFactoryMethod(ConstructorResolver.java:489) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.instantiateUsingFactoryMethod(AbstractAutowireCapableBeanFactory.java:1361) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBeanInstance(AbstractAutowireCapableBeanFactory.java:1191) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:563) at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:523) at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:339) at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:347) at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:337) at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:202) at org.springframework.beans.factory.support.DefaultListableBeanFactory.instantiateSingleton(DefaultListableBeanFactory.java:1155) at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingleton(DefaultListableBeanFactory.java:1121) at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:1056) at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:987) at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:627) at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:752) at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:439) at org.springframework.boot.SpringApplication.run(SpringApplication.java:318) at org.springframework.boot.SpringApplication.run(SpringApplication.java:1361) at org.springframework.boot.SpringApplication.run(SpringApplication.java:1350) at com.traitement.Traitement.TraitementApplication.main(TraitementApplication.java:11) 2025-04-23T16:06:01.901+01:00 WARN 19764 --- [Traitement] [ main] s.c.a.AnnotationConfigApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'step1' defined in class path resource [com/traitement/Traitement/config/BatchConfig.class]: Unsatisfied dependency expressed through method 'step1' parameter 0: No qualifying bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' available: expected at least 1 bean which qualifies as autowire candidate. Dependency annotations: {} 2025-04-23T16:06:01.920+01:00 INFO 19764 --- [Traitement] [ main] .s.b.a.l.ConditionEvaluationReportLogger : Error starting ApplicationContext. To display the condition evaluation report re-run your application with 'debug' enabled. 2025-04-23T16:06:01.966+01:00 ERROR 19764 --- [Traitement] [ main] o.s.b.d.LoggingFailureAnalysisReporter : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* APPLICATION FAILED TO START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Description: Parameter 0 of method step1 in com.traitement.Traitement.config.BatchConfig required a bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' that could not be found. Action: Consider defining a bean of type 'org.springframework.batch.core.configuration.annotation.StepBuilderFactory' in your configuration. Process finished with exit code 1

**ChatGPT a dit :**

L'erreur que vous rencontrez indique qu'il manque un bean de type StepBuilderFactory dans votre configuration Spring Batch. Ce bean est nécessaire pour la création des étapes dans le processus de batch. Voici comment vous pouvez corriger cette erreur.

**Étapes pour résoudre l'erreur**

1. **Ajouter le bean**StepBuilderFactory**à votre configuration.**

Dans votre classe de configuration BatchConfig, vous devez ajouter un bean pour StepBuilderFactory. Voici un exemple de configuration :

java

CopierModifier

import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { private final JobBuilderFactory jobBuilderFactory; private final StepBuilderFactory stepBuilderFactory; public BatchConfig(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory) { this.jobBuilderFactory = jobBuilderFactory; this.stepBuilderFactory = stepBuilderFactory; } @Bean public StepBuilderFactory stepBuilderFactory() { return new StepBuilderFactory(); } *// Votre méthode 'step1' devrait utiliser ce factory pour créer des étapes.* @Bean public Step step1() { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) .reader(personReader()) .processor(personProcessor()) .writer(personWriter()) .build(); } *// Ajoutez ici d'autres méthodes pour reader, processor, et writer* }

1. **Assurez-vous que la dépendance Spring Batch est bien incluse dans le**pom.xml.

Vérifiez que vous avez la bonne version de spring-batch-core dans votre fichier pom.xml. À l'intérieur de la section des dépendances, vous avez déjà un élément qui ressemble à ceci :

xml

CopierModifier

<dependency> <groupId>org.springframework.batch</groupId> <artifactId>spring-batch-core</artifactId> <version>5.0.0</version> </dependency>

Cette dépendance devrait être correcte, mais si vous avez des versions non compatibles avec Spring Boot 3.x, vous pourriez rencontrer des erreurs. Assurez-vous également que Spring Batch est bien configuré pour fonctionner avec la version de Spring Boot que vous utilisez (3.4.4).

1. **Vérifiez votre fichier**application.properties**ou**application.yml.

Assurez-vous que vos ressources d'entrée sont bien configurées. L'erreur FileNotFoundException indique que le fichier fichierV11.txt n'a pas été trouvé dans le chemin spécifié. Vous devez vous assurer que le fichier est situé dans le bon répertoire (src/main/resources/input/), ou ajuster le chemin dans votre configuration pour qu'il corresponde à l'emplacement correct.

1. **Structure du projet et ressource manquante** :

L'erreur liée au fichier fichierV11.txt semble indiquer que ce fichier n'est pas présent à l'emplacement attendu (src/main/resources/input/fichierV11.txt). Assurez-vous que ce fichier existe bien à cet emplacement. Si vous avez un autre emplacement pour ce fichier, vous devrez ajuster le chemin d'accès dans votre code.

**Résumé des changements :**

* Ajoutez un StepBuilderFactory comme bean dans votre configuration.
* Vérifiez que votre fichier de ressources (fichierV11.txt) est bien présent au bon endroit.
* Assurez-vous que les dépendances dans pom.xml sont compatibles avec Spring Boot 3.x.

Si vous appliquez ces modifications, votre application devrait démarrer correctement.

**Vous avez dit :**

donne moi un code corrigé

**ChatGPT a dit :**

Voici un exemple de code corrigé pour votre classe de configuration Spring Batch, prenant en compte la correction du problème du StepBuilderFactory manquant ainsi que le fichier fichierV11.txt manquant.

**Classe de Configuration Spring Batch (**BatchConfig.java**)**

java

CopierModifier

package com.traitement.Traitement.config; import com.traitement.Traitement.reader.PersonReader; import com.traitement.Traitement.writer.PersonWriter; import org.springframework.batch.core.Step; import org.springframework.batch.core.configuration.annotation.EnableBatchProcessing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.configuration.annotation.DefaultBatchConfigurer; import org.springframework.batch.core.launch.JobLauncher; import org.springframework.batch.core.launch.support.RunIdIncrementer; import org.springframework.batch.core.repository.JobRepository; import org.springframework.batch.core.StepExecutionListener; import org.springframework.batch.core.Job; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration @EnableBatchProcessing public class BatchConfig { private final JobBuilderFactory jobBuilderFactory; private final StepBuilderFactory stepBuilderFactory; public BatchConfig(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory) { this.jobBuilderFactory = jobBuilderFactory; this.stepBuilderFactory = stepBuilderFactory; } *// Assurez-vous que StepBuilderFactory est bien injecté* @Bean public StepBuilderFactory stepBuilderFactory() { return stepBuilderFactory; *// Pas besoin de redéfinir si déjà injecté* } *// Définir la méthode step1* @Bean public Step step1() { return stepBuilderFactory.get("step1") .<Person, Person>chunk(10) *// La taille du chunk, ajustez en fonction de vos besoins* .reader(personReader()) *// Le reader que vous définirez ailleurs* .processor(personProcessor()) *// Le processor que vous définirez ailleurs* .writer(personWriter()) *// Le writer que vous définirez ailleurs* .build(); } *// Définir le job* @Bean public Job job1(JobRepository jobRepository, Step step1) { return jobBuilderFactory.get("job1") .incrementer(new RunIdIncrementer()) .start(step1) .build(); } *// Méthode pour la configuration du reader* @Bean public PersonReader personReader() { return new PersonReader("src/main/resources/input/fichierV11.txt"); *// Assurez-vous que le fichier existe ici* } *// Méthode pour la configuration du processor* @Bean public PersonProcessor personProcessor() { return new PersonProcessor(); } *// Méthode pour la configuration du writer* @Bean public PersonWriter personWriter() { return new PersonWriter(); } }

**Code pour le Reader (**PersonReader.java**)**

Assurez-vous que le chemin du fichier est correct. Voici un exemple de la classe PersonReader qui pourrait être utilisée avec FileReader.

java

CopierModifier

package com.traitement.Traitement.reader; import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; import org.springframework.batch.item.ItemReader; import com.traitement.Traitement.model.Person; public class PersonReader implements ItemReader<Person> { private final String filePath; public PersonReader(String filePath) { this.filePath = filePath; } @Override public Person read() throws Exception, UnexpectedInputException, ParseException, NonTransientResourceException { try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) { String line; while ((line = reader.readLine()) != null) { *// Supposons que chaque ligne est une Person, ajustez cela en fonction de vos données* String[] data = line.split(","); return new Person(data[0], data[1]); *// Exemple de création d'objet Person* } } catch (IOException e) { throw new Exception("Erreur lors de la lecture du fichier", e); } return null; *// Si aucune ligne n'est lue* } }

**Assurez-vous que votre fichier de données existe dans**src/main/resources/input/fichierV11.txt**.**

**Code pour le**PersonWriter**(écrivain de données)**

Voici un exemple simple de PersonWriter qui pourrait être utilisé dans votre batch.

java

CopierModifier

package com.traitement.Traitement.writer; import java.util.List; import org.springframework.batch.item.ItemWriter; import com.traitement.Traitement.model.Person; public class PersonWriter implements ItemWriter<Person> { @Override public void write(List<? extends Person> items) throws Exception { *// Exemple de traitement : afficher dans la console* for (Person person : items) { System.out.println("Écriture de la personne : " + person); } } }

**Modèle**Person**(exemple simple)**

Vous devez également avoir une classe Person comme modèle pour vos objets batch.

java

CopierModifier

package com.traitement.Traitement.model; public class Person { private String firstName; private String lastName; public Person(String firstName, String lastName) { this.firstName = firstName; this.lastName = lastName; } *// Getters et setters* public String getFirstName() { return firstName; } public void setFirstName(String firstName) { this.firstName = firstName; } public String getLastName() { return lastName; } public void setLastName(String lastName) { this.lastName = lastName; } @Override public String toString() { return "Person{firstName='" + firstName + "', lastName='" + lastName + "'}"; } }

**Fichier**application.properties

Assurez-vous que le fichier fichierV11.txt existe bien dans le répertoire src/main/resources/input/ ou mettez à jour le chemin si nécessaire dans le code.

**Conclusion**

Avec ces modifications, vous avez ajouté le bean StepBuilderFactory correctement et configuré les composants nécessaires à Spring Batch. Vous devriez également être en mesure de résoudre l'erreur de chemin du fichier et démarrer votre application correctement.

**Vous avez dit :**

C:\Users\n.sahil\Downloads\Traitement\Traitement\src\main\java\com\traitement\Traitement\config\BatchConfig.java:58:77 java: illegal start of expression

**ChatGPT a dit :**

​

Haut du formulaire

Joindre

Rechercher

Raisonner

Bas du formulaire

ChatGPT peut faire des erreurs. Envisagez de vérifier les informations importantes.

?