


⚠ You signed in with another tab or window. [Reload](#) to refresh your session.

[Code](#)[Issues](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)[Insights](#) master ▾

...

[cours-springbatch](#) / EVAL.md

pjvilloud Create EVAL.md ✓

 1 contributor 145 lines (123 sloc) | 6.28 KB

...

communes dans un fichier plat

Le but est de créer un batch lisant les communes en base de données et produisant un fichier tel que celui-ci (exemple avec 3 communes en base):

```
Total codes postaux : 2
01300 - 01006 - Saint Ambleon : 45,74950 5,59432
01300 - 01454 - Virignin : 45,72674 5,71282
07460 - 07024 - Banne : 44,36078 4,15114
Total communes : 3
```

Les communes présentées dans le fichier sont ordonnées par code postal, puis par code Insee et les coordonnées GPS n'ont que 5 chiffres significatifs après la virgule. Le nombre total de codes postaux est affiché dans le premier enregistrement et le nombre total de communes est inscrit en dernière ligne.

Mise en place

- Créer un package `dbexport` dans `com.ipiecoles.batch`
- Créer une classe `CommunesDBExportBatch` contenant le code ci-dessous. Cette classe contiendra les différents beans nécessaires à votre job.



You signed in with another tab or window. [Reload](#) to refresh your session.

```
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
@Configuration
public class CommunesDBExportBatch {

    @Bean
    @Qualifier("exportCommunes")
    public Job exportCommunes(){
        return null;
    }

}
```

- Créer un fichier `application.properties` dans `src/test/resources` et y ajouter `spring.batch.job.enabled=false` pour éviter le lancement des batchs lors de l'exécution du test
- Créer le package `com.ipiecoles.batch.utils` dans `src/test/java` et y ajouter la classe `BatchTest`

```
package com.ipiecoles.batch.utils;
import org.springframework.batch.core.Job;
import org.springframework.batch.core.launch.JobLauncher;
import org.springframework.batch.core.repository.JobRepository;
import org.springframework.batch.test.JobLauncherTestUtils;
import org.springframework.beans.factory.annotation.Autowired;
public class BatchTest {

    @Autowired
    protected JobLauncher jobLauncher;
    @Autowired
    protected JobRepository jobRepository;
    protected JobLauncherTestUtils jobLauncherTestUtils;
    protected void initializeJobLauncherTestUtils(Job job) {
        this.jobLauncherTestUtils = new JobLauncherTestUtils();
        this.jobLauncherTestUtils.setJobLauncher(jobLauncher);
        this.jobLauncherTestUtils.setJobRepository(jobRepository);
        this.jobLauncherTestUtils.setJob(job);
    }

}
```

- Créer la classe de test `CommunesDBExportBatchIntegrationTest` dans `src/test/java` package `com.ipiecoles.batch`

⚠ You signed in with another tab or window. [Reload](#) to refresh your session.

```
import com.ipiecoles.batch.repository.CommuneRepository;
import com.ipiecoles.batch.utils.BatchTest;
import org.junit.After;
import org.junit.Assert;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.batch.core.*;
import org.springframework.batch.test.AssertFile;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import java.io.File;
import java.util.Date;
@ExtendWith(SpringExtension.class)
@SpringBootTest
public class CommunesDBExportBatchIntegrationTest extends BatchTest {

    @Autowired
    @Qualifier("exportCommunes")
    private Job exportCommunes;
    @Autowired
    private CommuneRepository communeRepository;
    @BeforeEach
    @After
    public void setupAndTeardown(){
        communeRepository.deleteAll();
        this.initializeJobLauncherTestUtils(exportCommunes);
    }

    @Test
    public void testSimpleJobOk() throws Exception {
        //Given
        JobParametersBuilder paramsBuilder = new JobParametersBuilder();
        paramsBuilder.addDate("date", new Date());
        paramsBuilder.addString("filePath", "target/test.txt");
        JobParameters jobParameters = paramsBuilder.toJobParameters();
        communeRepository.save(new Commune("01006", "Saint Ambleon", "01300", 45.
        communeRepository.save(new Commune("01454", "Virignin", "01300", 45.72673
        communeRepository.save(new Commune("07024", "Banne", "07460", 44.36077827
        // when
        JobExecution jobExecution = jobLauncherTestUtils.launchJob(jobParameters)
        JobInstance actualJobInstance = jobExecution.getJobInstance();
        ExitStatus actualJobExitStatus = jobExecution.getExitStatus();
        // then
        Assert.assertEquals("exportCommunes", actualJobInstance.getJobName());
        Assert.assertEquals(ExitStatus.COMPLETED, actualJobExitStatus);
        AssertFile.assertFileEquals(new File("src/test/resources/laposte_out_test
    }
```



You signed in with another tab or window. [Reload](#) to refresh your session.

- Créer ou modifier le fichier `src/test/resources/laposte_out_test.txt` pour qu'il contienne le contenu suivant

```
Total codes postaux : 2
01300 - 01006 - Saint Ambleon : 45,74950 5,59432
01300 - 01454 - Virignin : 45,72674 5,71282
07460 - 07024 - Banne : 44,36078 4,15114
Total communes : 3
```

Quelques informations supplémentaires :

- Une première step devra récupérer le nombre de codes postaux distincts (cf la méthode `countDistinctCodePostal` de `CommuneRepository`) et le nombre total de communes et mettre ces informations à disposition des prochaines steps.
- La step d'écriture du fichier sera idéalement codée avec un `RepositoryItemReader` (exemple <https://www.programmersought.com/article/71013801913/>) à défaut un `JpaPagingItemReaderBuilder` .
- Il est nécessaire d'utiliser un `FlatFileItemWriter` pour l'écriture du fichier plat (plus d'infos <https://pjvilloud.github.io/cours-springbatch/#/springbatch-chunk-item-writer2>)
- Il faudra utiliser le principe des Header/Footer pour la première et la dernière ligne (plus d'infos <https://docs.spring.io/spring-batch/docs/current/reference/html/common-patterns.html>)
- Pour le formatage des lignes des communes, utiliser un `FormatterLineAggregator` à la place du `DelimitedLineAggregator` du cours et spécifier le format `"%5s - %5s - %s : %.5f %.5f"`
- Le fichier généré doit être placé dans le dossier `target` et s'appeler `test.txt`