Spring Batch Tutorial: Lectura de información de un archivo de Excel

Petri Kainulainen 20 de marzo de 2016 comentarios 28

Spring Batch , Spring Framework

Es bastante fácil crear un trabajo de Spring Batch que lea sus datos de entrada desde un CSV o un archivo XML porque estos formatos de archivo son compatibles de manera inmediata.

Sin embargo, si queremos leer los datos de entrada de nuestro trabajo por lotes de un archivo .XLS o .XLSX que se creó con Excel, tenemos que trabajar un poco más. Esta publicación de blog nos ayuda a resolver este problema.

Empecemos.

Si no está familiarizado con Spring Batch, **debe leer** las siguientes publicaciones en el blog antes de seguir leyendo esta publicación de blog:

- Spring Batch Tutorial: Introducción especifica el término trabajo por lotes, explica por qué debe usar Spring Batch e identifica los componentes básicos de un trabajo de Spring Batch.
- Tutorial de Spring Batch: Obtener las dependencias requeridas con Maven

Tutorial de Spring Batch: Obtener las dependencias requeridas con Gradle

pendencias Spring Batch con Gradle.

ación de un archivo describe cómo puede leer

ML.

cación de ejemplo

rios trabajos de Spring Batch que procesan la información del estudiante de un curso en línea. Esta vez necesitamos crear un trabajo de Spring Batch que pueda importar información del estudiante desde un archivo de Excel. Este archivo contiene una lista de estudiantes que proporciona la siguiente información para nuestra aplicación:

- El nombre del estudiante.
- La dirección de correo electrónico del estudiante.
- El nombre del paquete comprado.

Cuando leemos la información del alumno de un archivo de Excel, tenemos que transformar esa información en objetos de *StudentDTO* que procesamos en nuestro trabajo por lotes. La clase *StudentDTO* contiene la información de un solo alumno, y su código fuente se ve de la siguiente manera:

```
1  public class StudentDTO {
2
3      private String emailAddress;
4      private String name;
5      private String purchasedPackage;
6
7      public StudentDTO() {}
8
9      public String getEmailAddress() {
10         return emailAddress;
11     }
12
13     public String getName() {
14         return name;
15     }
dieciséis
17     public String getPurchasedPackage() {
18         return purchasedPackage;
19     }
20
21     public void setEmailAddress(String emailAddress) {
22         this.emailAddress = emailAddress;
```

```
23 }
```



```
ng name) {

Package(String purchasedPackage) {
e = purchasedPackage;
```

r de elementos que lea la información del
nemos que agregar algunas dependencias
ión.

Obteniendo las Dependencias Requeridas

Si queremos leer los datos de entrada de nuestro trabajo Spring Batch desde un documento de Excel, tenemos que agregar las siguientes declaraciones de dependencia en nuestro script de compilación:

- **Spring Batch Excel** es una extensión Spring Batch que proporcionaimplementaciones de *ItemReader* para Excel. Lamentablemente, en este momento, la única forma de obtener el archivo jar requerido es compilarlo desde la fuente .
- **Apache POI** proporciona una API de Java para documentos de Microsoft Office. Es una dependencia opcional de Spring Batch Excel, y podemos usarla para leer datos de entrada de documentos .XLS y .XLSX.

Supongo que la mayoría de ustedes no quiere construir Spring Batch Excel desde la fuente. Es por eso que lo hice por ti. Eche un vistazo a las aplicaciones de ejemplo de esta publicación de blog: Spring example y Spring Boot example . Estas aplicaciones tienen secuencias de comandos de compilación de Gradle y Maven que utilizan el repositorio de Maven encontrado en el directorio de *repositorio* .

Lectura adicional:

- Spring Batch Tutorial: Obtener las dependencias requeridas con Maven
- Spring Batch Tutorial: Obtener las dependencias requeridas con Gradle
- Spring Batch Extensions: construyendo desde el origen

ones de dependencia requeridas en nuestro script
gurar el *lector de elementos* que puede leer la
a de cálculo de Excel.

n archivo de Excel

e estudiantes de nuestro curso. Este archivo se
leta es: *data / students.xlsx* . El contenido de esta
hoja de cálculo de Excel se ve de la siguiente manera:

```
NAME            |EMAIL_ADDRESS               |PURCHASED_PACKAGE
Tony Tester     |tony.tester@gmail.com       |master
Nick Newbie     |nick.newbie@gmail.com       |starter
Ian Intermediate|ian.intermediate@gmail.com  |intermediate
```

Nuestra hoja de cálculo no contiene caracteres de tubería ('|'). Este carácter se usa aquí para separar las diferentes celdas entre sí.

Como ya sabemos, podemos proporcionar datos de entrada para nuestro trabajo por lotes Spring configurando un bean *ItemReader* . Podemos configurar un bean *ItemReader* , que lee la información del *estudiante del* archivo *students.xlsx* , siguiendo estos pasos:

1. Cree una clase *ExcelFileToDatabaseJobConfig* y *anótelo* con la anotación *@Configuration* . Esta clase es la clase de configuración de nuestro trabajo por lotes y contiene los beans que describen el flujo de nuestro trabajo por lotes.
2. Cree un método que configure nuestro bean *ItemReader* y asegúrese de que el método devuelva un objeto *ItemReader <StudentDTO>* .
3. Implemente el método creado siguiendo estos pasos:
    1. Cree un nuevo objeto *PoiItemReader <StudentDTO>* .
    2. Asegúrese de que el lector creado ignore el encabezado de nuestra hoja de cálculo.
    3. Configure el lector creado para leer la información del *alumno del* archivo *data / students.xlsx* que se encuentra en classpath.

ar una información del estudiante fila en una

apperRowMapper clase. Esta clase rellena los

eado mediante el uso de los nombres de columna

nuestra hoja de cálculo.

creado <*StudentDTO*> .

tabaseJobConfig se ve de la siguiente manera:


```
atch.item.ItemReader;
atch.item.excel.RowMapper;
atch.item.excel.mapping.BeanWrapperRowMapper;
atch.item.excel.poi.PoiItemReader;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.env.Environment;
import org.springframework.core.io.ClassPathResource;

@Configuration
public class ExcelFileToDatabaseJobConfig {

    @Bean
    ItemReader<StudentDTO> excelStudentReader() {
        PoiItemReader<StudentDTO> reader = new PoiItemReader<>();
        reader.setLinesToSkip(1);
        reader.setResource(new ClassPathResource("data/students.xlsx"));
        reader.setRowMapper(excelRowMapper());
        return reader;
    }

    private RowMapper<StudentDTO> excelRowMapper() {
        BeanWrapperRowMapper<StudentDTO> rowMapper = new BeanWrapperRowMapper<>
        rowMapper.setTargetType(StudentDTO.class);
        return rowMapper;
    }
}
```

Este enfoque funciona siempre que nuestra hoja de cálculo de Excel tenga una fila de encabezado y los nombres de columna de la fila de encabezado se puedan resolver en los nombres de campo de la clase *StudentDTO* .

Sin embargo, es muy posible que tengamos que leer los datos de entrada de una hoja de cálculo que no tiene una fila de encabezado. Si este es el caso, tenemos que crear un *RowMapper* personalizado que transforma las filas de nuestra hoja de cálculo en objetos de *StudentDTO* .

Podemos crear un *RowMapper* personalizado siguiendo estos pasos:

1. Crea una clase *StudentExcelRowMapper* .

> y pase el tipo de objeto creado (*StudentDTO*)

wSet rowSet) de la *interfaz RowMapper <T>*

) .

l objeto creado. Podemos leer los valores de
ando el *método getColumnValue (int*
t . Además, debemos recordar que el índice de la

3. Devuelve el objeto *StudentDTO* creado .

El código fuente de la clase *StudentExcelRowMapper se* ve de la siguiente manera:

```
1  import org.springframework.batch.item.excel.RowMapper;
2  import org.springframework.batch.item.excel.support.rowset.RowSet;
3
4  public class StudentExcelRowMapper implements RowMapper<StudentDTO> {
5
6      @Override
7      public StudentDTO mapRow(RowSet rowSet) throws Exception {
8          StudentDTO student = new StudentDTO();
9
10         student.setName(rowSet.getColumnValue(0));
11         student.setEmailAddress(rowSet.getColumnValue(1));
12         student.setPurchasedPackage(rowSet.getColumnValue(2));
13
14         return student;
15     }
dieciséis }
```

Después de que hemos creado nuestro asignador de fila personalizado, debemos realizar los siguientes cambios en la configuración de nuestro bean *ItemReader* :

1. Asegúrese de que nuestro *lector de elementos* **no ignore** la primera línea de los datos de entrada.
2. Reemplace el antiguo método *excelRowMapper ()* con un método que devuelva un nuevo objeto *StudentExcelRowMapper* .

Después de realizar estos cambios en la clase *ExcelFileToDatabaseJobConfig* , su código fuente se ve de la siguiente manera:

```
1  import org.springframework.batch.item.ItemReader;
2  import org.springframework.batch.item.excel.RowMapper;
3  import org.springframework.batch.item.excel.poi.PoiItemReader;
4  import org.springframework.context.annotation.Bean;
```

```
5 import org.springframework.context.annotation.Configuration;
ore.io.ClassPathResource;


baseJobConfig {


xcelStudentReader() {
tDTO> reader = new PoiItemReader<>();
ew ClassPathResource("data/students.xlsx"));
excelRowMapper());


tDTO> excelRowMapper() {
elRowMapper();
```


Lectura adicional:

- Spring Batch Excel: Configuración
- Spring Batch Excel: RowMappers

Vamos a resumir lo que aprendimos de esta publicación de blog.

Resumen

Esta publicación de blog nos ha enseñado cuatro cosas:

- Si queremos leer los datos de entrada de un trabajo de Spring Batch desde una hoja de cálculo de Excel, tenemos que agregar Spring Batch Excel y las dependencias de POI de Apache en nuestro script de compilación.
- Si queremos leer los datos de entrada utilizando Spring Batch Excel y Apache POI, tenemos que usar la clase *PoiItemReader* .
- Podemos mapear las filas de nuestra hoja de cálculo en objetos *T* utilizando la *clase BeanWrapperRowMapper* <*T*> siempre que nuestra hoja de cálculo Excel tenga una fila de encabezado y los nombres de columna de la fila de encabezado se puedan resolver en los nombres de campo de la clase *T.*
- Si nuestra hoja de cálculo de Excel no tiene una fila de encabezado o los nombres de columna de la fila de encabezado no pueden resolverse en los nombres de campo de la

onente mapeador de fila personalizado que

r> .

emplo de esta publicación de blog de Github:

ARTÍCULOS RELACIONADOS

SPRING FRAMEWORK /

Pruebas unitarias de Spring MVC Controllers: API REST

SPRING FRAMEWORK /

Crear URLs RESTful con Spring MVC 3.1 Parte tres: UrlRewriteFilter

SPRING FRAMEWORK /

ámicamente la ruta del recurso? En lugar de codificar la ruta del archivo en el lector.
Normalmente esto sería dinámico, podría tener un sitio web donde las personas suben archivos y deben procesarse.

RESPUESTA

gtvk

28 de diciembre de 2016, 22:36

Gracias Petri, estoy enfrentando un problema similar al descrito por Amos. Necesita pasar un archivo de forma dinámica en lugar de codificarlo con dificultad. La ayuda sería apreciada

¡Gracias!

RESPUESTA

kishore

15 de febrero de 2017, 12:54

Proporcione el código de muestra para el ejemplo anterior ...

Gracias.

RESPUESTA

el código de la ruta de recursos dinámicos para mavera.

Hola,

No sé cómo puede usar un "nombre de archivo dinámico" y usar Spring Batch Excel. Si escribe el suyo `ItemReader`, puede implementar la lógica de lectura de archivos como desee, pero esto significa que también debe escribir el código que lee el contenido del archivo de entrada.

Una opción es que usted pueda crear su propio `Resource`archivo que pueda ubicar el archivo de entrada de forma dinámica, pero no estoy seguro si esto es posible y cómo debería implementarse.

RESPUESTA

Praveen

3 de abril de 2017, 16:07

Hola,

Tengo un archivo de Excel que contiene rowspan y colspan. ¿Puedes compartir un ejemplo en el que podamos leer los datos del intervalo desde el archivo Excel?

RESPUESTA

Excel usando lotes de primavera. He guardado solo
ıt.xlsx. Por primera vez funciona bien, pero la
el trabajo me está dando la excepción
lice de la hoja (1) está fuera del rango (0..0)'.
ı. ¿Puedo saber de dónde está aumentando el

REPLY

Petri

May 30, 2017, 22:35

Hmm. I think that you have just found a bug from my example. I will take a closer look at it.

REPLY

Daniel Henao

September 9, 2017, 00:09

Hey, have you fixed it? sorry I Ask but it happes to me too

REPLY

Petri

September 20, 2017, 23:25

Ah. I forgot to report my findings. I am sorry about that :(

In any way, it seems that this problem is caused by the Spring Batch Excel. The problem is that it doesn't reset the number of the current sheet after it has

neans that when the job is run for the second time,
ot be opened (because it doesn't exist).

problem by cloning the Spring Batch Excel
uired change to the `AbstractExcelItemReader`
e required change, you have to create new jars
rs that are provided by this example.

Yohan

July 20, 2017, 08:49

Do you have an example where the Excel file is being submitted as Multipartfile.? Your example reads the excel file from the classpath and sets the resource on the Item reader. I am hoping for an example where the job is triggered when a user sends a request with the excel file payload as such open an input stream read the excel. If you we are reading the excel file as input stream, how do we set the resource on the reader bean?

REPLY

Petri

July 23, 2017, 21:36

Unfortunately I don't have an example that reads the Excel file from an HTTP request. However, you could save the uploaded file to a directory and read the file from the upload directory by using Spring Batch. Actually, I will add this example to my to-do list and I maybe I will implement it after I have released my testing course.

REPLY

Yohan

September 6, 2017, 07:09

Have you had a chance to implement the uploading Excel file with HTTP request?

y time to write that example (I am still recording my

REPLY

Geek

February 7, 2018, 14:42

Did you get a chance to implement the uploading Excel file with HTTP request?I googled so many things but not finding any proper solution.

Petri

February 11, 2018, 10:11

Hi,

I am sorry that it took me some time to answer to your comment. As you probably guessed, I am still recording my testing course. That being said, the course should be finally done after a few weeks, and I can concentrate on writing more content to my blog.

About your problem: I haven’t been able to find a proper solution to it because Spring Batch doesn’t provide a good support for reading data from files that are determined at runtime. I assume that it’s possible to support this use case, but I just haven’t found the solution yet.

CHUA KWAN LIANG

August 28, 2017, 10:52

ticular sheet? how to do it in reader ya? Thank

[illegible] `PoiItemReader` and `AbstractExcelItemReader` classes, and it seems that there is no way to set the opened sheet (or at least I couldn't find it). In other words, if you want to do this, yo have to make the required changes directly to the source code of Spring Batch Excel.

REPLY

Daniel Henao

September 12, 2017, 00:47

The main classes of PIO have that option. but the Extension of spring batch doesnt. you just set the sheet you want using .getSheet(X)

REPLY

---

bala

January 27, 2018, 09:36

I have an requirement where i need to read dynamic excel sheets and process same and write in db. All of the examples i have seen so far requires a dto to access the excel file.Is there anyway i can read excel sheet dynamically without dto ?

REPLY

mathieu

April 4, 2018, 10:47

:he same problem

You have to implement a custom `ItemReader` that reads the input file by using Apache POI. Unfortunately I cannot give you the exact steps because they depend from the structure of the input file.

REPLY

Naren

March 7, 2018, 14:11

I have added spring batch excel dependencies to my build script but I am getting error : import org.springframework.batch.item.excel.* cannot be resolved.

REPLY

Petri

March 12, 2018, 10:54

Are you using Maven or Gradle? Also, did your IDE reload the dependencies of your project after you made changes to your build script?

REPLY

Messi

092/how-to-import-from-xls-to-mysql-database-

Hi This helped me a lot. Unfortunately facing some issues.

excelStudentReader works fine for the firsttime. When i make a call again with the same excel, getting below error

Caused by: java.lang.IllegalArgumentException: Sheet index (3) is out of range (0..2)

I tried changing the below line in PoiItemReader getSheet()

return new PoiSheet(this.workbook.getSheetAt(sheet)); to return new PoiSheet(this.workbook.getSheetAt(0));

This time it worked but not as expected.

Can you please look into it.

REPLY

Petri

May 9, 2018, 13:34

Hi,

Este es un error conocido de la biblioteca Spring Batch Excel . Parece que el proyecto original ha sido abandonado y alguien ha creado un tenedor que corrige este problema

actualizar mis ejemplos de Spring Batch durante

e problema cuando lo haga.

¿CÓMO ESCRIBIR MEJORES PRUEBAS?

Si tiene dificultades para escribir pruebas automáticas que **acepten el cambio** , debe averiguar cómo mi curso de evaluación puede ayudarlo a **escribir pruebas para las aplicaciones Spring y Spring Boot** .

Guarde mi nombre, correo electrónico y sitio web en este navegador para la próxima vez que comento.

ENVIAR

PUBLICACIÓN ANTERIOR: JAVA TESTING WEEKLY 11/2016
SIGUIENTE PUBLICACIÓN: JAVA TESTING WEEKLY 12/2016

¿CÓMO ESCRIBIR MEJORES PRUEBAS?

Si tiene dificultades para escribir pruebas automáticas que **acepten el cambio** , debe averiguar cómo mi curso de evaluación puede ayudarlo a **escribir pruebas para las aplicaciones Spring y Spring Boot** .

ESCRIBIR MEJORES PRUEBAS

Prueba con curso de primavera

Java Testing Weekly

JUnit 5 Tutorial

Tutorial de prueba Spring MVC

Tutorial de WireMock

Escribir pruebas limpias

Escritura de pruebas para el código de acceso a datos

MASTER SPRING FRAMEWORK

Tutorial de Spring Data JPA

Tutorial de Spring Data Solr

Primavera de las trincheras

Tutorial de prueba Spring MVC

Tutorial Social de Primavera

Usando jOOQ con Spring

CONSTRUYE TU APLICACIÓN

Comenzando con Gradle

Tutorial de Maven

ES

¿CÓMO ESCRIBIR MEJORES PRUEBAS?

Si tiene dificultades para escribir pruebas automáticas que **acepten el cambio** , debe averiguar cómo mi curso de evaluación puede ayudarlo a **escribir pruebas para las aplicaciones Spring y Spring Boot** .

DESDE EL BLOG

Reciente Popular Favoritos

Java Testing Weekly 40/2018

Java Testing Weekly 39/2018

Java Testing Weekly 38/2018

Java Testing Weekly 37/2018

Java Testing Weekly 36/2018

¿CÓMO ESCRIBIR MEJORES PRUEBAS?

Si tiene dificultades para escribir pruebas automáticas que **acepten el cambio** , debe averiguar cómo mi curso de evaluación puede ayudarlo a **escribir pruebas para las aplicaciones Spring y Spring Boot** .