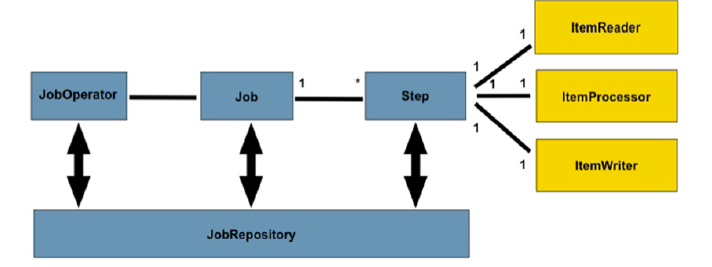
前言: 基本上batch機制有很多種實作,cron job,quartz….

Java ee 7 也終於推出自己的實作

(看來有參考到spring…….spring batch 2.0)

Batch application for the java platform (JSF 352)



架構解說

1. 一個job基本上可以有很多的步驟

One job to many steps

Job start=>step1=>step2=> decision control =>step 6=> job end

1. 每個step基本上 如果type是chunk都會有 reader/processor/writer

Reader=>processor=>writer

基本上看起來都必須根據規範去實作對應的reader/processor/writer

1. Chunk 這邊是指

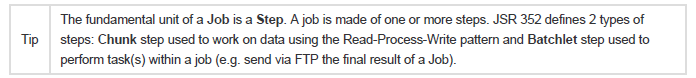
每個step 都會有reader/processor/writer

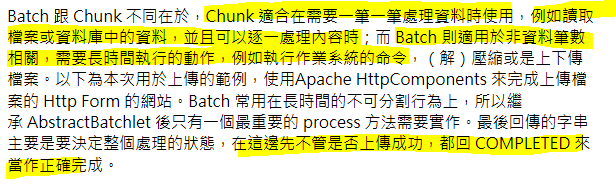
視為一個chunk

(這邊是spring batch的觀念拿來用)

1. Step 基本上有兩種type

* Chunk: reader-process-write (這樣步驟 就一個transation)
* Batchlet: tasks with a job (全部通通一個transation搞定)





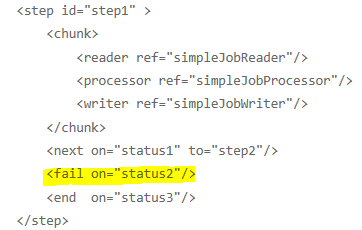


1. Job基本上透過xml設定去決定step如何運行,該如何跑,

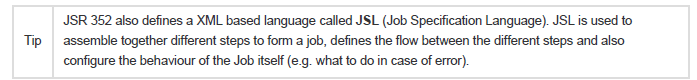
簡單的設定如下



複雜的情況如下(如果有錯誤的話 或是甚麼情況 該跑去哪個step)

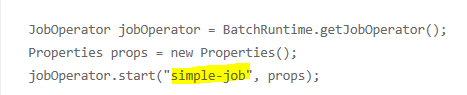


使用XML定義 call JSL



1. jobOperator基本上控制每個job 啟動/重啟/結束 相關設定處理

以下是啟動的情況



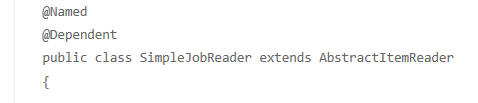
Start(xxxx,props) xxxx這邊是指xml檔案名稱

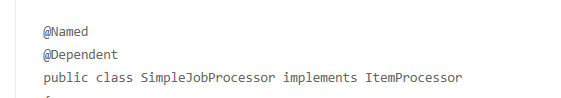


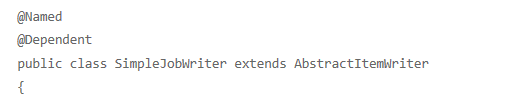
1. JobRepository 基本上控管job執行狀態 現在 及過去執行情況

可以實作by db/fileSystem

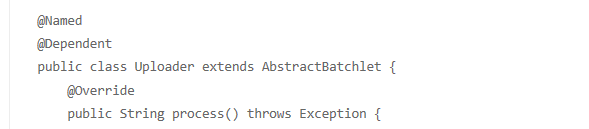
1. Reader/processor/writer需分別繼承 實作 相關類別



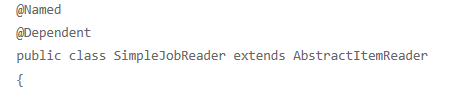




Batchlet實作如下

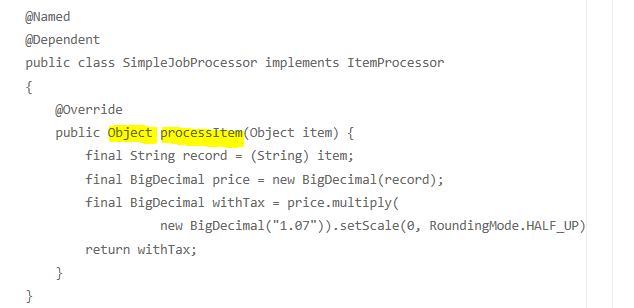


1. Reader 實作如下

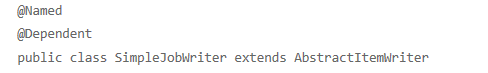




1. Processor實作如下



1. Writer實作如下





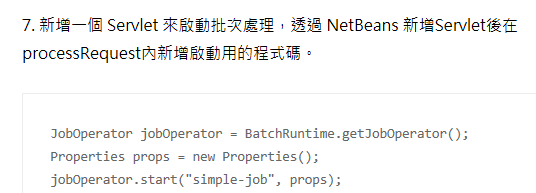
1. 寫job的xml描述檔

放在scr/META-INF/batch-jobs/下

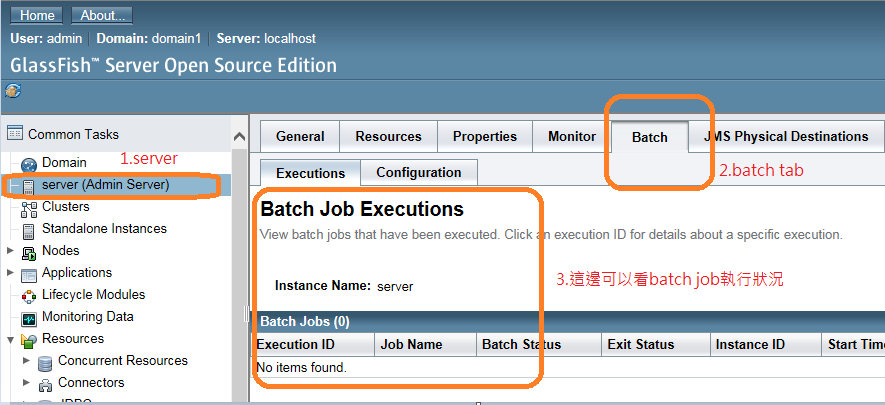
PS.放在sub folder下面會被忽略



1. 程式呼叫啟動這個job

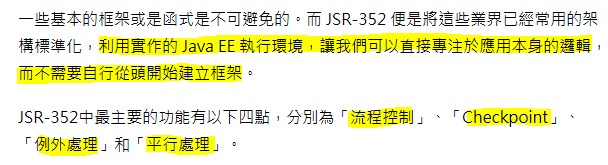


1. Glassfish可以看batch job執行狀況

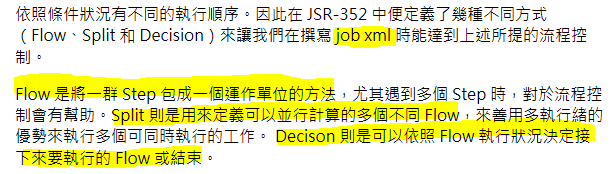


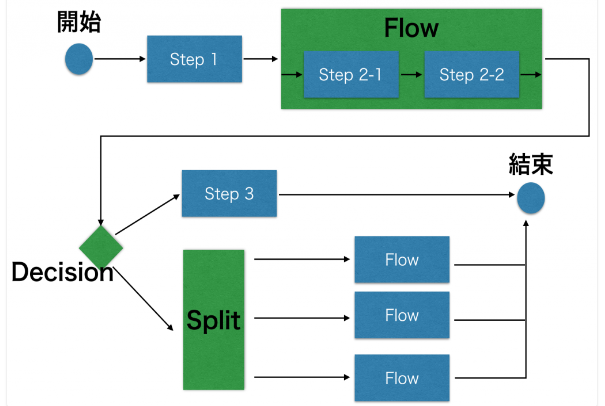
1. Java batch job 重點

流程控制/checkpoint/例外處理/平行處理

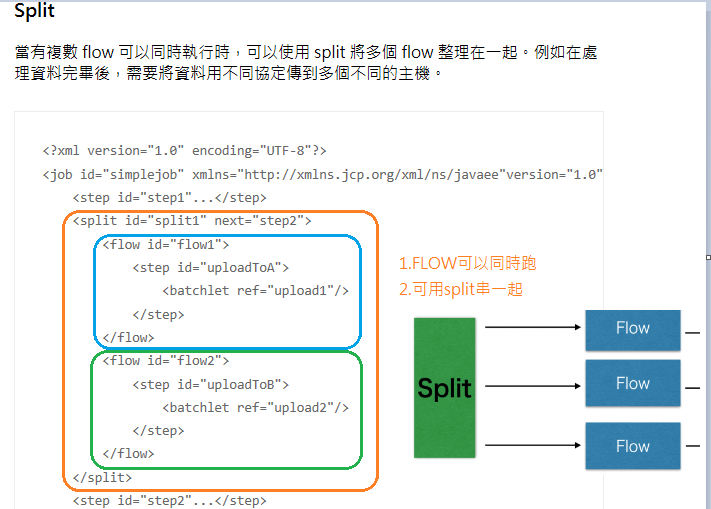


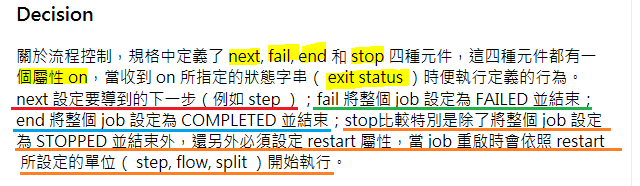
* 流程控制













狀態如何控制(batchlet return string

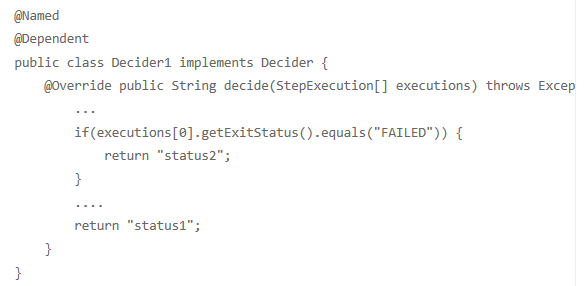
, chunk use stepContext.setExistStatus)



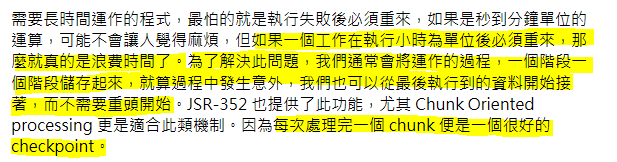
額外利用decision來控制流向(VS 上面是step內部控管)



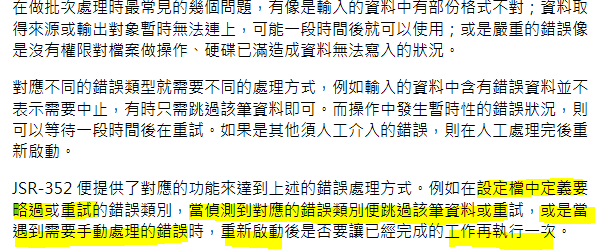
Decider需自己實作



* Checkpoint

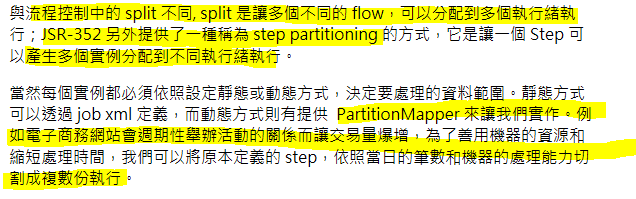


* 例外處理



1. 可定義錯誤類別 略過 or 重試
2. 錯誤失敗 可以選擇重頭跑 或是 上次錯的地方繼續跑

* 平行處理(這邊看起來也蠻強大 值得研究一下)



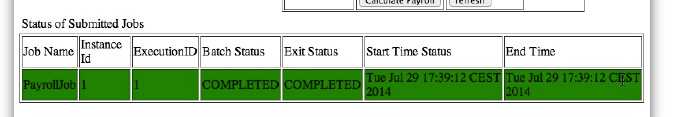
* checkpoint



1. 取得job執行流程狀態

Servlet內 呼叫JOB時 監控狀態



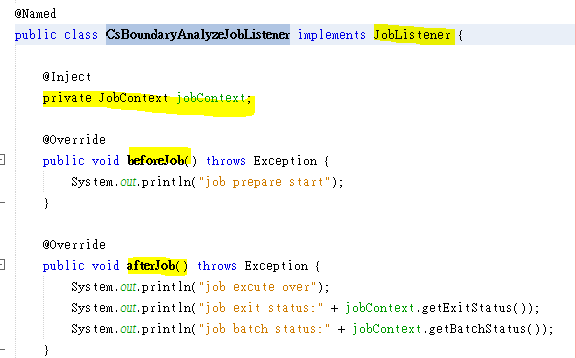


1. Listener

可以分別針對job做listener

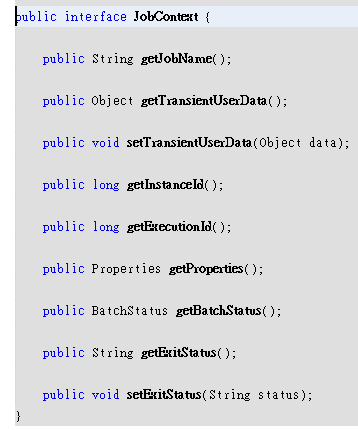
或者是針對step做listener

分別實作如下圖



jobContext可以透過注入 得到狀態

相關屬性如下

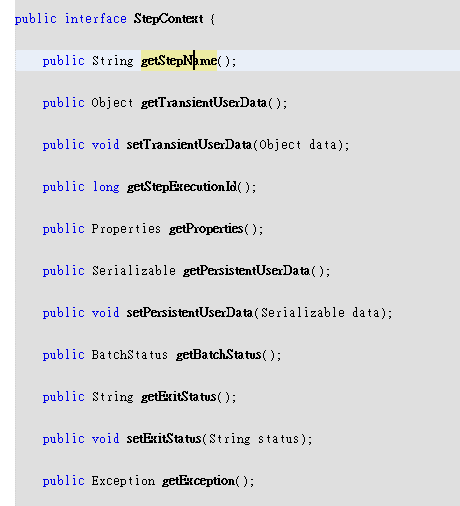


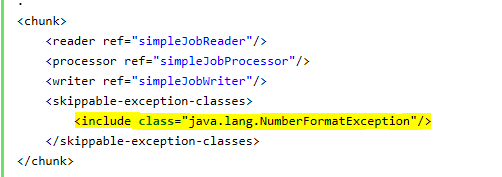
Step也可以有listener



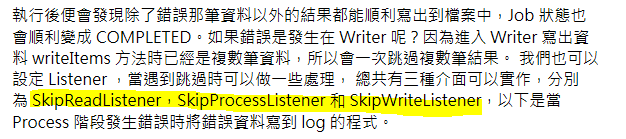
stepContext也可以透過注入方式取得

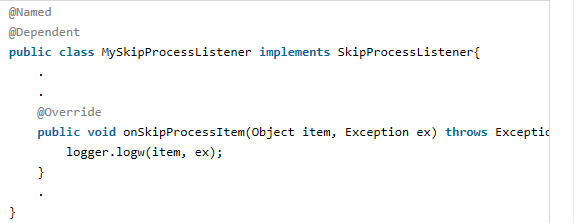
stepContext相關資訊 如下圖



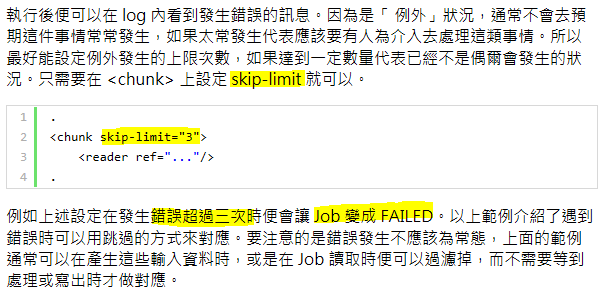
1. 例外處理(跳過/重試/重跑)
2. 跳過有問題的資料

可以針對read/process/write 分別去寫listener

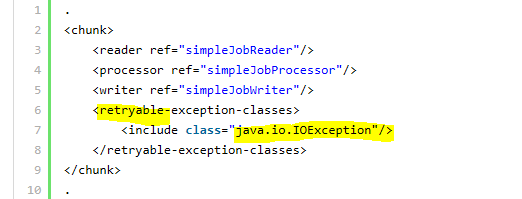




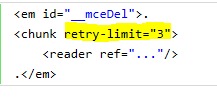




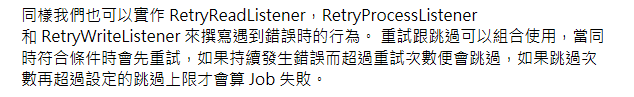
1. 重試



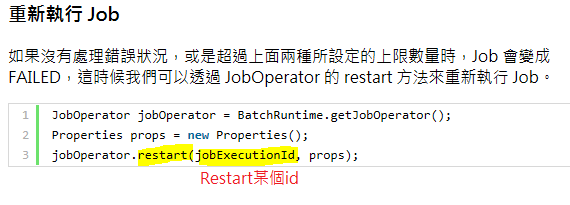
可以加上重試次數限制

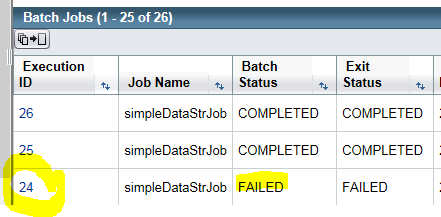


也可以實作retry 相關的read/process/write listener



1. 重跑





會去取得之前執行情況 從錯誤的step 開始跑

Ex. Step1/step2/step3/step4

如果1,2成功

3錯誤

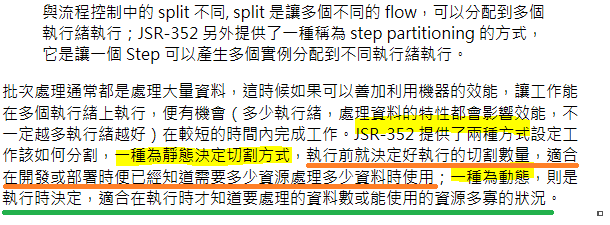
Job跳掉

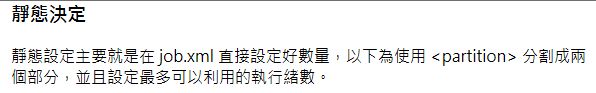
重新執行會從3開始

重跑如果希望重頭來



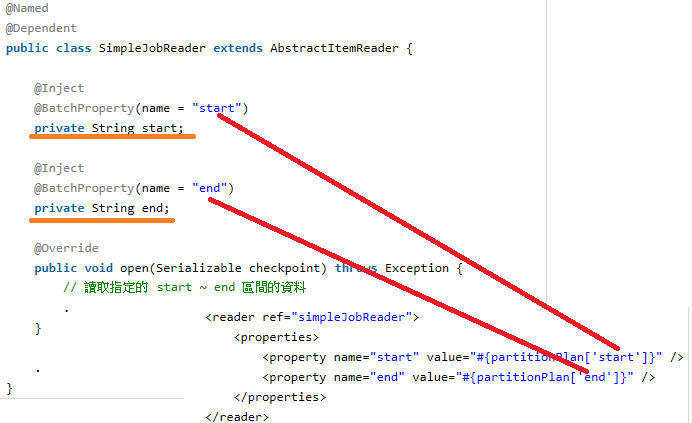
1. 平行處理

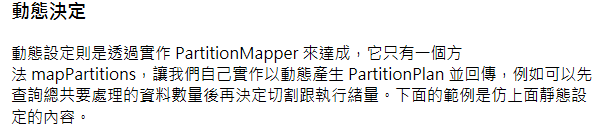




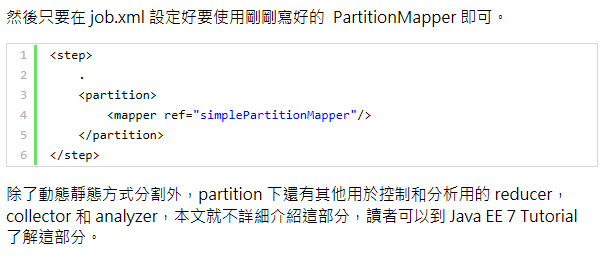


讀取的地方 透過參數 去抓 到哪要處理start/end資料









<<<SAMPLE 參考範例 搭配現行情況>>>

<<<

下面範例就是 我撈出list data

然後一一把資料 變成字串 屬性用”/”串起來

最後再輸出List<String>

>>>

1. 基本上reader可以inject facade

\*Readitem要注意 這邊最後只能回傳一個object

因此即使open 或 init method可以一次撈全部

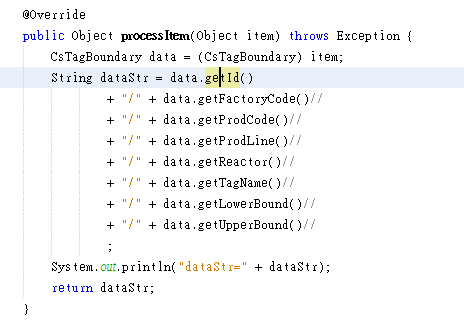
Readitem method也必須要每次只能回傳一個object



1. Processor

<<<注意這邊一次一個物件 是從readitem過來>>>

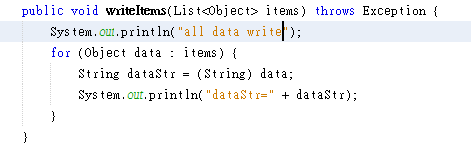
<<然後也是一次只回一個物件出去>>



1. Writer

<<< 這邊注意 writeitems 是收到list<Object> >>>

<<< list<object>> 其實是從processor 收集而來 >>



以上流程可以如下描述

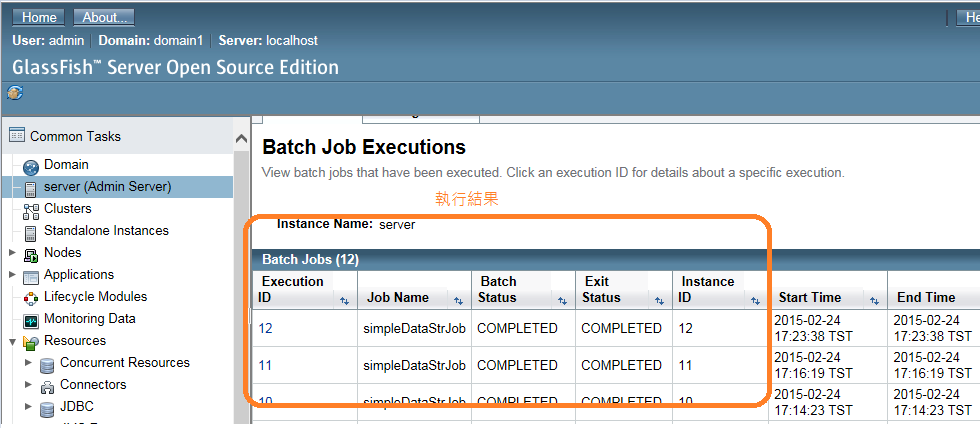
下圖範例是我準備數量三的list 然後去看reader/processor/writer的關係

基本上就是

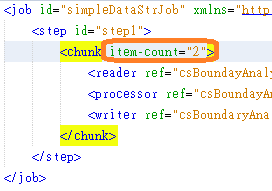
(Read a,process a),(read b,process b),(read c,process c),最後整個write list出去



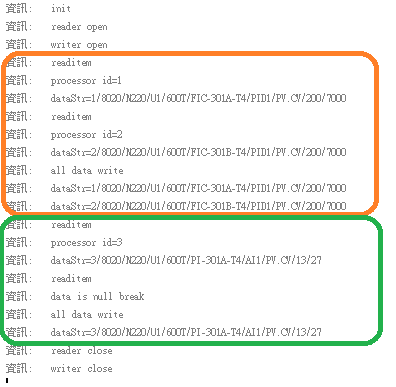
Glassfish server看batch執行結果



下圖我改chunk 兩個一組



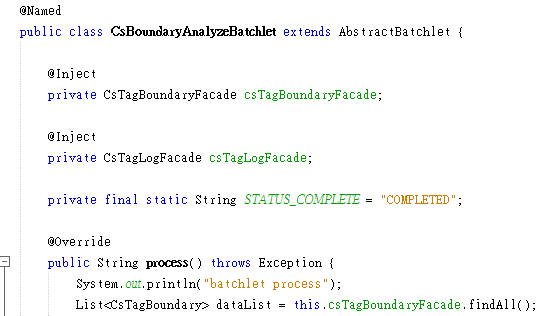
執行結果<<< 三筆資料就拆成兩塊輸出了 >>>



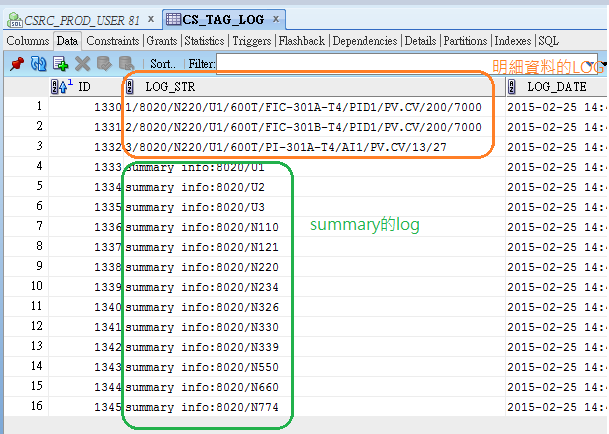
接下來我這邊使用batchlet來做summary的處理



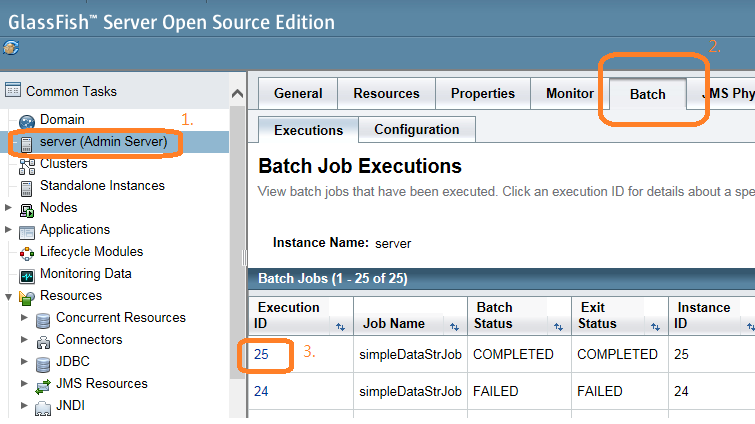
Batchlet如下

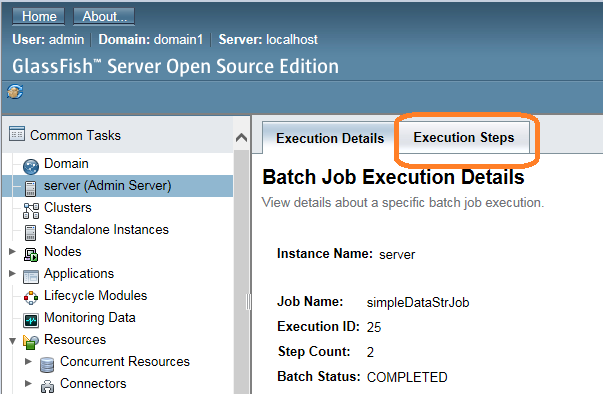


最後我DB LOG TABLE 就會像



Glassfishs console batch可以點進去看 detail





Detail畫面如下



接下來測試

錯誤 重跑處理

\*經測試 重跑處理 可以區分

-1.錯誤的step重跑

Step1(run ok)=>step2(error) 重開會是從step開始 如果沒有刻意設定的話

-2.錯誤的step裡面 從失敗的區塊重新跑(chunk mode only)

假設step1使用chunk mode

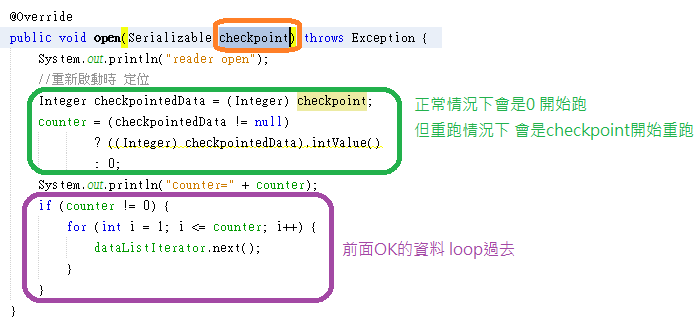
預設情況下如果第27筆故意錯

則step1失敗 如果又沒有刻意控管資料的話 會有20筆資料輸出

因為chunk item-counter預設10

重跑的話 會從step1 資料第21筆開始跑資料

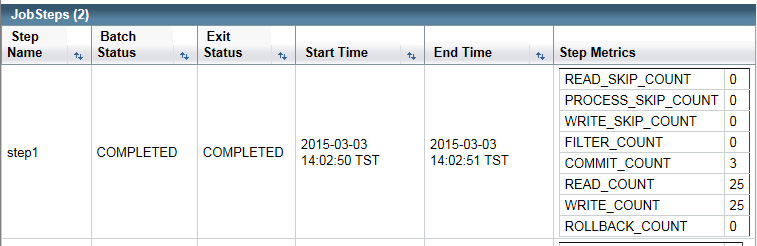
但 讀取資料的部分 要記得使用 checkpoint 處理



失敗點的紀錄



重跑之後step1紀錄



看起來45筆資料 分成 第一次錯(但寫了20筆) + 第二次重跑(寫後面的25筆)

(補充參考材料)

1. Java ee 7 其實取自spring batch 2.0的Chunk Oriented 處理方式

<https://blog.codecentric.de/en/2013/07/spring-batch-and-jsr-352-batch-applications-for-the-java-platform-differences/>

1. Spring 針對batch處理事實上分成兩種

Chunk 及 TaskletStep

基本上 看起來 很簡單的處理 直接用Tastle模式處理即可

<http://wuaner.iteye.com/blog/1768351>

<http://www.onlinetechvision.com/taskletstep-oriented-processing-in-spring-batch/>

Tasklet基本上設計看起來就是在一個transation搞定

所以沒有transation fail問題

1. Operator 的java doc

<http://docs.oracle.com/javaee/7/api/javax/batch/operations/JobOperator.html>

1. Java batch xml設定 參考

<http://docs.oracle.com/javaee/7/tutorial/batch-processing004.htm#BCGDDBBG>

1. Chunk 預設item-count 是10

所以每十筆 write一次

1. Batch 狀態如下

