

批次介紹

什麼是批次？


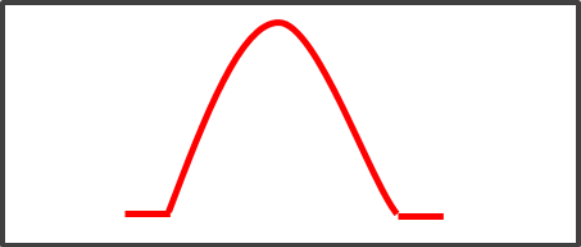
- 分批讀取及執行
- 有固定的規則及行為
- 在有限資源下，可以降低系統的負擔

批次與模組

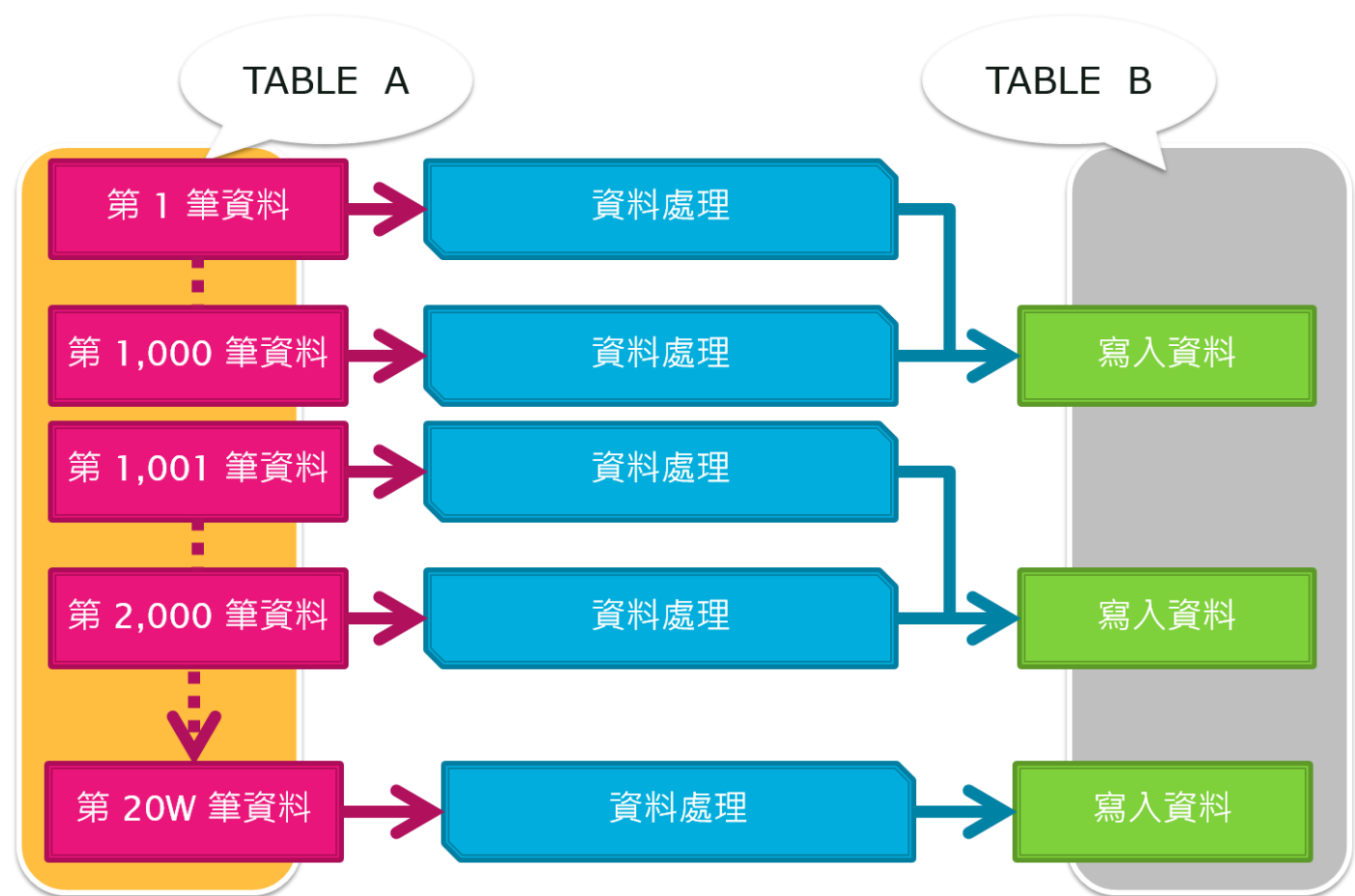
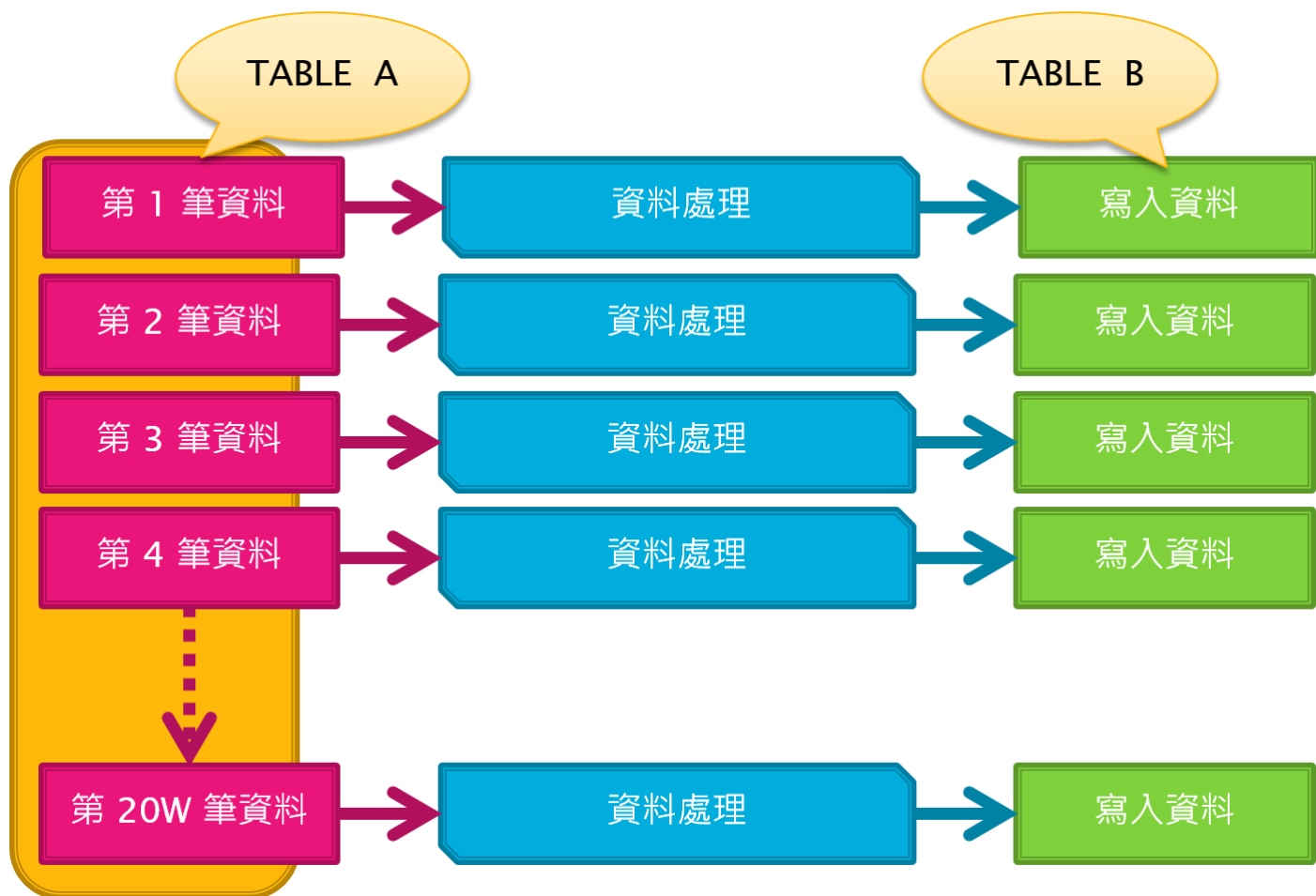
相同

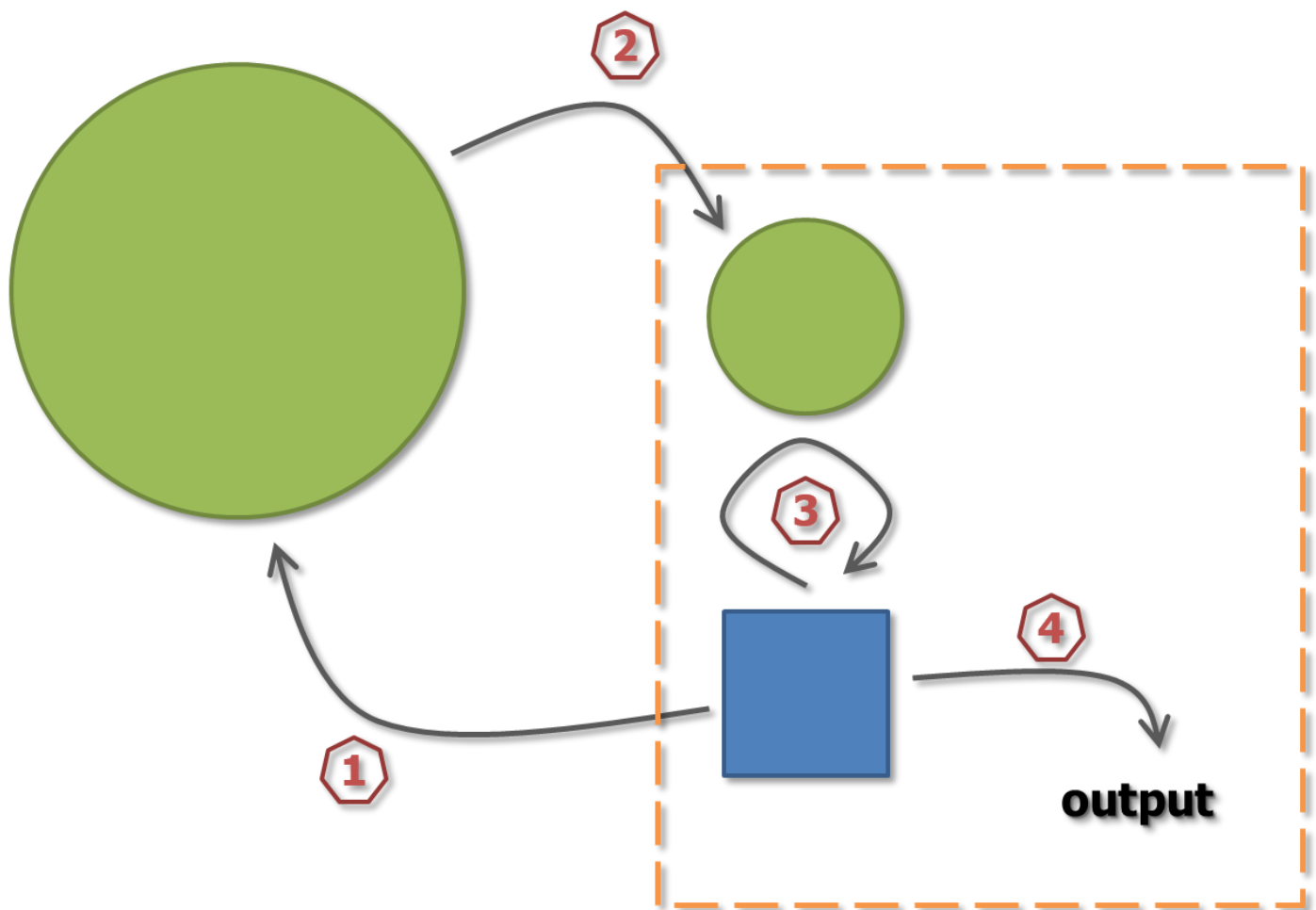
- 可寫邏輯判斷
- 可讀寫資料庫
- 可讀寫檔案

相異

	Batch	Module
查詢物件	BatchQueryDataSet	DataSet
更新物件	BatchUpdateDataSet	DataSet
記憶體使用量		

範例





1. 去資料庫查詢資料
2. 只取回要處理的那一批資料
3. 逐筆處理資料
4. 輸出處理結果

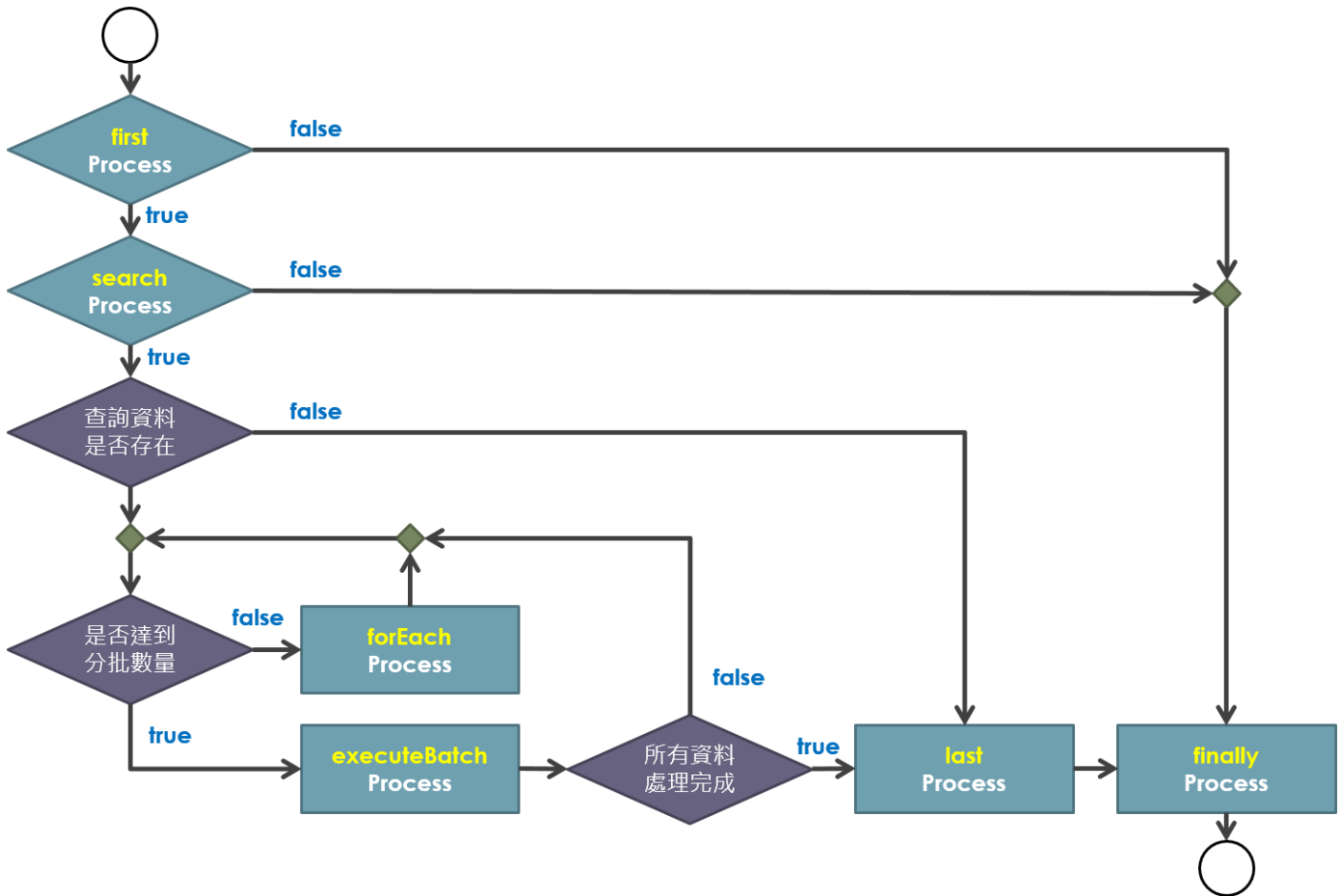
批次的種類

1. Table→Table
2. Table→File
3. File→Table
4. File→File

程式架構

新批次架構 (BatchConstructor)

流程圖



BatchConstructor

`int deleteTable(String tableSchema, String tableName, Map<String, Object> queryParams, int executeStatusType)`

- 刪除資料表格
- tableSchema：欲刪除之資料表格的 schema。
- tableName：欲刪除之資料表格的 table name。
- queryParams：刪除資料的條件。
- executeStatusType：刪無資料時視為正常與否。
- 目標表格必須要有 PK。
- 所帶入之條件僅以「=」去查詢。
- 於資料庫 DP 下之 Table 無法使用此方法。

`BatchUpdateDataSet getBatchUpdateDataSet(String SQL, int executeStatusType)`

- 取得 BatchUpdateDataSet
- SQL：當 `Options.setUseDynamicSQLByUpdate` 傳入 `false` 時，則使用靜態的 SQL，當傳入 `true` 時，則使用動態 SQL。預設為 `false`。

- `executeStatusType`：更新無資料時視為成功與否，以及新增重複時視為成功與否。

`void createCountType(String countType)`

- 建立件數紀錄類型
- `countType`：欲建立的件數紀錄類型名稱。

`void addCountNumber(String countType, int countNumber)`

- 依指定件數紀錄類型加入件數
- `countType`：欲累加的件數紀錄類型。
- `countNumber`：欲累加的數值。

`void addErrorLog(Object message, Object memo)`

- 加入錯誤紀錄
- `message`：訊息。
- `memo`：摘要。

`int writeErrorLog()`

- 寫出錯誤紀錄，回傳 `ErrorLog` 筆數

`Map<String, Integer> getAllCountTypeAndNumber()`

- 取得所有件數紀錄

`Options getOptions()`

- 可取得 `BatchConstructor` 的 `Options` 物件，進行批次執行時之相關設定。
- 需在 `execute` 執行前設定好屬性。每次 `execute` 完畢後 `Options` 物件會被初始化，所有設定回復至預設值。

`execute(new BatchConstructor.DataBaseHandler() {});`

- 處理資料來源為「資料庫」之類別。

`execute(new BatchConstructor.FileHandler() {});`

- 處理資料來源為「檔案」之類別。

p.s. 一個 `execute` 可視作一個批次，整支批次其實就是在覆寫這兩個類別中的方法

Options

`void setUseDynamicSQLByQuery(boolean isDyna)`

- 查詢是否使用動態 SQL，預設為 `false` 不使用。

`void setUseDynamicSQLByUpdate(boolean isDyna)`

- 異動是否使用動態 SQL，預設為 `false` 不使用。

void setGroupKeys(String[] groupKeys)

- 群組資料鍵值。
- 注意：資料來源請自行先下 ORDER BY。

void setTerminated(boolean isTerminated)

- 異動失敗時是否終止，預設為 false 不中止。
- 注意：當一批資料錯誤時，可決定要繼續往下做(會 redo 一次)或是終止，但前一批已經 commit 之資料已無法 rollback。

void setIndividualStatistics(boolean isIndividual)

- 是否使用個別統計，預設為 false 不使用。
- 當有多條 buds 時可開啟此選項，每條連線將獨立計算輸出件數與錯誤件數。

void setFile(File inputFile)

- 輸入檔案，請配合 FileHandler 使用。

void setSysCodeForQuery(String sysCodeForQuery)

- 取得獨立連線，傳入使用查詢之子系統。Ex：DS_ZZ 請傳入 ZZ。

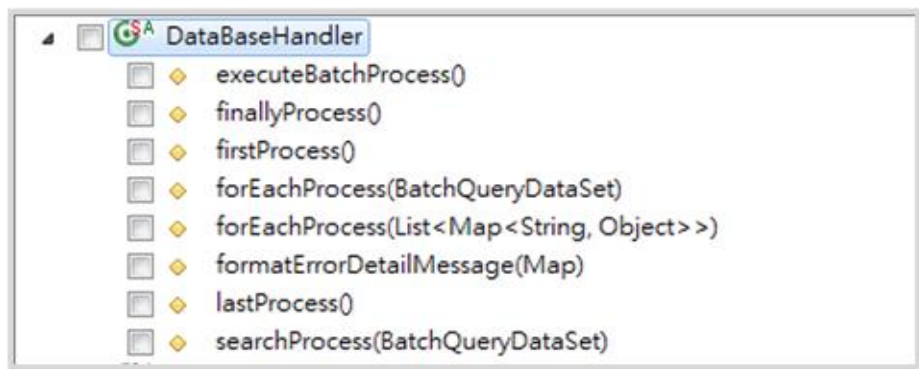
void setSysCodeForUpdate(String sysCodeForUpdate)

- 取得獨立連線，傳入使用異動之子系統。EX：DS_EP 請傳入 EP。
- 注意：BatchConstructor 之特性為，當 buds 開始交易控制時，Transaction.begin 亦同時被啟動。若要跳脫此特性，可利用此設定強制 buds 取獨立連線。

void setKeepCountManager(boolean keepCountManager)

- 是否保留計數統計到批次結束後，預設為 false 不保留。
- 計數器在進入 finallyProcess 之前即會被清空，若欲在 finallyProcess 中取得統計件數，請設定為 true。

DataBaseHandler



boolean firstProcess()

- 執行於此批次開始時，若 return false 則跳至 finallyProcess，若 return true 則繼續執行 searchProcess。

`boolean searchProcess(BatchQueryDataSet bqds)`

- 供查詢程序使用，若 `return false` 則跳至 `finallyProcess`，若 `return true` 則開始進行分批，逐筆處理資料。

`void forEachProcess(BatchQueryDataSet bqds)`

`void forEachProcess(List<Map<String, Object>> dataList)`

- 逐筆處理資料，若有給定群組鍵值則應覆寫第二個方法，所傳入參數為整群相同鍵值之 `List`；若無則應覆寫第一個方法。

`void executeBatchProcess()`

- 每批執行一次，執行時機先於批次指令送至資料庫執行之前。

`void lastProcess()`

- 資料來源皆處理完畢後執行此方法，若前述過程發生錯誤時此方法不會被執行。

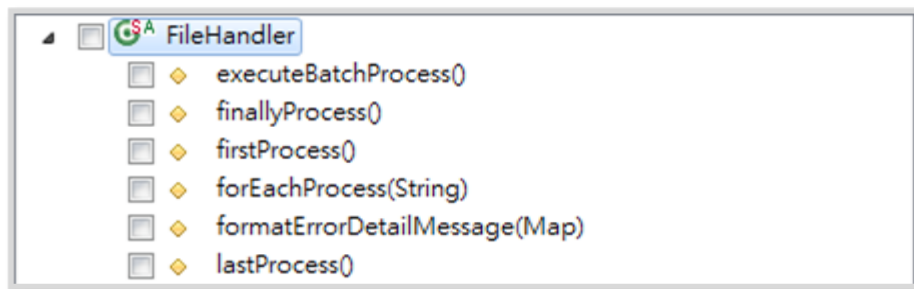
`void finallyProcess()`

- 批次結束時必定執行，只要 `bc.execute` 被呼叫，則必定會執行此方法。

`String formatErrorDetailMessage(Map errorDataMap)`

- 異動資料庫錯誤時會呼叫此方法，傳入之參數為組成 `Map` 後的各筆錯誤內容，因此異動錯誤次數會等於此方法被呼叫之次數。可自行定義回傳之錯誤訊息寫入 `ErrorLog` 中。

FileHandler



`boolean firstProcess()`

- 執行於此批次開始時，若 `return false` 則跳至 `finallyProcess`，若 `return true` 則繼續執行。

`void forEachProcess(String str)`

- 逐行處理資料，每讀取檔案的一行文字則呼叫此方法一次，傳入參數即為所讀取到之文字內容。

`void executeBatchProcess()`

- 每批執行一次，執行時機先於批次指令送至資料庫執行之前。

`void lastProcess()`

- 資料來源皆處理完畢後執行此方法，若前述過程發生錯誤時此方法不會被執行。

`void finallyProcess()`

- 批次結束時必定執行，只要 `bc.execute` 被呼叫，則必定會執行此方法。

`String formatErrorMessage(Map errorDataMap)`

- 異動資料庫錯誤時會呼叫此方法，傳入之參數為組成 `Map` 後的各筆錯誤內容，因此異動錯誤次數會等於此方法被呼叫之次數。可自行定義回傳之錯誤訊息寫入 `ErrorLog` 中。

[例] `BatchConstructor_Sample.java`

[例] `BatchConstructor_Sample2.java`

[練習]

原始批次架構

繼承類別

- 需要繼承子系統的 BatchBean，並實作 execute method

使用類別

BatchQueryDataSet

- 批次查詢資料庫

BatchUpdateDataSet

- 批次新增或更新資料庫

CountManager

- 紀錄設定的件數

ErrorLog

- 記錄錯誤訊息

ErrorHandler

- 依錯誤的層級取得錯誤的資料

GroupHandler

- 同一批多個 table 異動
- 移除錯誤的資料，並將正確的資料寫入

BatchProcessor

- 將正確的資料重新執行異動

程式架構

```
try{
    // 從資料庫取的資料
    // 逐筆處理資料
}catch(Exception e){
    // 錯誤處理
}finally{
    // 最後要執行的步驟
}
```

交易控制

- 以批為單位
- 不同批的資料無法同時 commit 或 rollback
- 只能在批與批中間決定後續資料是否要繼續處理
- 查無資料或資料重複可以透過設定來決定是否算異常

[例] Batch_Sample.java

與 BatchConstructor 的比較

```
public void execute(DataBaseHandler handler) throws Exception {
    try {
        handler.firstProcess();
        handler.searchProcess(bqds);
        while (currentIndex <= totalCount) {
            buds.beginTransaction();
            try {
                while (bqds.next()) {
                    handler.forEachProcess(bqds);
                }
                handler.executeBatchProcess();
                executeBatch(buds, handler);
            } catch (Exception e) {
                buds.rollbackTransaction();
            } finally {
                addCountNumber(ERROR_COUNT, errorCount);
            }
        }
        handler.lastProcess();
    } finally {
        handler.finallyProcess();
    }
}
```

[練習]

Online-Submit Batch 流程



```
ControlMCaller aControlMCaller = new ControlMCaller(true);  
String[] batchArgArray = new String[] { BAL_DATE };  
aControlMCaller.runBatch(batchId, execUnit, batchArgArray, true);
```

1. **batchId** : 批次的程式代號。
2. **execUnit** : 執行單位，若限定只能被一個單位執行則傳入 **null**。
3. **batchArgArray** : 執行批次時的傳入參數。

```
ZZ_X0Z004 theZZ_X0Z004 = new ZZ_X0Z004(batchId);  
ReturnMessage rm = new ReturnMessage();  
theZZ_X0Z004.startBatch(rm);  
if (rm.getReturnCode() != ReturnCode.OK) {  
    String errorMsg = "新增批次監控檔發生錯誤";  
    setExitCode(ERROR);  
    log.fatal(errorMsg);  
    throw new ModuleException(errorMsg);  
}
```

ZZ_X0Z004 於建構時可傳入單一參數 **batchId**，限定只能被一個單位執行；或是 **batchId** 與 **execUnit** 可允許不同單位同時執行。



```
if (args == null || args.length != 9 || args[0].indexOf("CTM") != -1) {  
    args = new ZZ_X0Z007(batchId).retrieveParam(null, 9);  
}  
if (args == null || args.length != 9) {  
    throw new ErrorInputException("傳入參數錯誤");  
}
```

注意：此為範例程式碼，實際情況請自行調整。為因應接收由 ControlM 排程帶起之傳入，以及由 online-submit 傳入之參數，程式碼略有不同。

```
ZZ_X0Z004 theZZ_X0Z004 = new ZZ_X0Z004(batchId);  
ReturnMessage rm = new ReturnMessage();  
theZZ_X0Z004.endBatch(rm);  
if (rm.getReturnCode() != ReturnCode.OK) {  
    String errorMsg = "刪除批次監控檔發生錯誤";  
    setExitCode(ERROR);  
    log.error(errorMsg);  
    throw new ModuleException(errorMsg);  
}
```

ZZ_X0Z004 於建構時可傳入單一參數 batchId，限定只能被一個單位執行；或是 batchId 與 execUnit 可允許不同單位同時執行。
注意：請確實刪除監控檔，否則將無法再帶起批次。

```

public void execute(String[] arg0) throws Exception {
    final BatchConstructor bc = new BatchConstructor("JARIOG001", "RIG1_B020", "日");

    //透過線上批次執行模組執行
    //宣告批次開始 (ZZ_X0Z004)

    try {

        //取得參數 (ZZ_X0Z007)
        //批次程式碼

    } catch (Exception e) {
        log.fatal("執行時發生錯誤", e);
        setExitCode(ERROR);
    } finally {
        //透過線上批次執行模組執行
        //宣告批次結束 (ZZ_X0Z004)
        printExitCode(getExitCode());
    }
}

```

批次作業執行狀況維護模組(ZZ_X0Z004)

void orderBatch(ReturnMessage rm, DataSet ds)

- order 該批次，但實際上批次主程式尚未被呼叫。
- 若該批次已被相同的單位 order 了，則本次的 order 動作將會失敗。
- 此動作會新增一筆進批次作業執行狀況檔，啟動時間為 null 的紀錄。
- 此 method 通常只會被 ControlMCaller 呼叫。

void startBatch(ReturnMessage rm)

- 設定批次的狀態為啟動(update 啟動時間)。
- 此 method 通常只會被批次主程式呼叫。

void endBatch(ReturnMessage rm)

- 設定批次的狀態為已結束(刪除該筆狀態)。
- 此 method 通常只會被批次主程式呼叫。

線上批次參數傳送模組(ZZ_X0Z007)

void setParam(int seq, String value, DataSet ds)

- 設定欲傳給批次的參數。
- seq：從 0 開始，對應到批次 String[] args 裡的傳入參數 index。重覆設定將會出現錯誤。
- value：設定 String[] args 傳入參數的值。
- 此 method 通常交由 ControlMCaller 處理。

String[] retrieveParam(String[] batchArgs)

`String[] retrieveParam(String[] batchArgs, int paramsCnt)`

- 若傳入的 `args` 是有值的(不是 `null`)，則直接回傳此 `args`。反之則從 DTZZX007 取得從前端設定的參數。取得後清空 DTZZX007 裡該 `batchId` 對應的參數資料。
- `paramsCnt`：取得參數數量，若數目不符合將取不到。
- 此 `method` 通常由批次主程式呼叫。

開發注意事項

- 盡量避免在批次迴圈內呼叫模組進行 I/O 處理
- 清空 Table 時可用 `com.cathay.common.util.EmptyTableHelper`
- 程式結束時要手動關閉連線
- 無法使用 Two Phase Commit