DCIM Chassis 管理的自動化整合測試

專題編號：111-CSIE-S017
執行期限：110 年第 1 學期至 111 年第 1 學期
指導教授：劉建宏
專題參與人員： 108820021 李以謙
108820031 簡上博

一、摘要

本專題的目標是要完成DCIM中管理Chassis的程式碼的整合測試，該程式碼由於效能不佳且分層架構不明確需要進行refactor，因此我們使用黑箱測試並蒐集效能相關資料來協助負責 refactor 的工程師，確保修改後功能仍完善。我們在TestNG 的框架下進行測試，並利用VirtualBox 模擬 Linux 系統以建立資料庫，使用 Postgres 和 DBeaver 來操作、查看資料庫。團隊採取 Scrum 開發模式，定時檢視開發進度及公司需求，動態調整達到最好的成果。

關鍵詞：business layer 和 data layer 的整合測試、Scrum、黑箱測試、測試連接資料庫

二、緣由與目的

我們參與了 Sunbird 和北科合作的產學計畫與實習；Sunbird 的產品是 DCIM 軟體 (Data Center Infrastructure Management Software:資料中心基礎設施管理軟體)，而公司的程式碼中，管理 Chassis 的部分被發現對效能有所影響，且 business layer 和 data layer 並未分離，因此決定進行 refactor 以改善系統效能、框架。我們進行黑箱測試並盡可能讓測試滿足所有條件(讓測試的 coverage 達到 100%)來確保 refactor 後功能的正確性;同時也蒐集與效能相關的資料以及定期向工程師詢問使用測試的情況來協助 refactor 工作的進行。

三、研究範圍

主要研究 business layer 和 data layer 的測試如何撰寫、將測試檔案連結到資料庫並 insert 測試資料、理解 method 的目的以及學習安裝和設定需要的環境、工具。

四、使用技術方法

(一) 使用 PgAdmin 匯入公司提供的資料庫 schema 到虛擬機中的資料庫，並自行將測試所需要的資料建立出來。
(二) 採取黑箱測試，使用 Assert Equal 來驗證程式結果，以及使用 debugger 和其他工具來觀察程式運行、所寫測試涵蓋的範圍及程式運行時間。
(三) 程式碼耦合性過高，method 環環相扣影響範圍甚廣，因此難以理解，我們會適當的做筆記釐清程式脈絡。
(四) 測試 Java 的程式效能，使用了 Jprofiler 來觀察 CPU 的運作。
(五) 若遇到不確定、無法解決的疑題，例如：系統環境設定、虛擬機的操作，亦或公司內部流程，我們會請教公司的工程師協助解決問題。

五、架構流程

(一) 團隊作業流程
我們的開發模式參考敏捷框架的 Scrum，此流程有助於預期時間成本，也能適時檢視團隊進度，讓不符合預期的地方能夠即時修正，以下是我們的開發流程：
1. Sprint 開始時，會創建 story，每個 story

用 point 估計工作量，以此推估完成所需時間。
2. 每週與教授討論目前的進度及作業方向，以確保內容的正確性。
3. Sprint 的最後一天會與公司開會檢視當次 Sprint 完成的內容是否符合預期，動態調整團隊作業以符合公司方的需求。
4. Sprint 最後會進行 Sprint retrospective，團隊成員提出當次 Sprint 做的好及可以改進的地方，增加團隊生產力。

(二) 測試撰寫流程
1. 為了決定工作的優先度，我們將 method 以是否嚴重影響效能分成兩部分。
2. 撰寫程式前，盡可能短時間完成第一次 trace code，其目的在於理解 method 的作用，並準備第一筆測試資料。
3. 使用測試資料跑 debug、coverage 工具，驗證自己的想法是否與程式運作相符；與此同時，決定下一個測試的範圍及測試資料。
4. 重複步驟三，直到 coverage 涵蓋所有待測 method。
5. 在本地跑 integration test，以確保測試 merge 進 production code 後能順利運行。
6. 發 PR，請求 merge 進入主程式。
7. 使用 JProfiler 測試 refactor 前後 method 的效能。
8. 整理 JProfiler 的結果，觀察執行時間差之原因，並將其傳給負責 refactor 的工程師，確認結果符合工程師的預期。

六、工具說明

(一) IntelliJ：Java 的開發環境工具，以此進行程式的撰寫、debug、查看 coverage。
(二) TestNG：Java 的測試框架。
(三) Maven：軟體專案管理及自動構建工具，主要用於環境的建構。
(四) PgAdmin：PostgreSQL 管理工具，以此連線資料庫，對資料庫進行 restore 等操作。
(五) DBeaver：管理資料庫的工具，以此查看資料庫之內容。
(六) JProfiler： Java 的效能分析工具，以此分析效能提供 refactor 前的效能資訊。

七、結論

該專案中，business layer 和 data layer 並未分離，因此資料庫的運作也包含於測試，若是在每個測試執行前都注入測試資料(BeforeMethod)，執行時間會非常久 (約 13 分鐘)，而先將全部資料一同注入資料庫(BeforeClass)，雖大幅降低測試執行時間(約 2 分鐘)，但執行時間仍遠超過沒有連接資料庫的測試。若是將 bussiness layer 與 data layer 分離，測試可以分開撰寫，邏輯測試的執行時間能夠大幅減少，由此可知分層架構對軟體維護的重要性。

在使用 Jprofiler 測試 method 在 refactor 前後的效能後，可以觀察到，有一些因為效能而做的 refactor 會使程式的執行時間下降，但也可以發現有部分程式修正了架構，卻導致執行時間變長（因為原本直接到資料庫拿資料，現在需要藉由 DAO 層從資料庫取資料），因此程式也需要在效能和維護成本中做出取捨。

除此之外，透過這次專題，我們還學習到如何妥善的規劃進度及基本的職場開發流程，更熟悉資料庫的操作。

參考文獻

[1] 李浩宇 Alex (2021)「【Java 技術指南】「TestNG 專題」單元測試框架之 TestNG 使用教程指南」取自：https://iter01.com/613558.html

[2] Kevin Wu (2018) “什麼是 Scrum？不是工程師也能懂的 Scrum 入門介紹！” 取自: https://reurl.cc/RrzKbr

[3] 苦中乐 (2018) “Intellij IDEA 集成 JProfiler 性能分析神器” 取自：https://blog.csdn.net/wytocsdn/article/details/79258247