## From Gossip@Openhome

# 非關語言:設計模式

程式設計是思維具體化的一種方式，是思考如何解決問題的過程，設計模式是在解決問題的過程中，一些良好思路的經驗集成，最早講設計模式，人們總會提到 **Gof** 的著作，它最早將經典的 23 種模式集合在一起說明，對後期學習程式設計，尤其是對從事物件導向程式設計的人們起了莫大的影響。後來設計模式一詞被廣泛的應用到各種經驗集成，甚至還有反模式（AntiPattern），反模式教導您如何避開一些常犯且似是而非的程式設計思維。

這邊的話將整理一些設計模式學習心得，實作的部份是使用 Java 與 Python，在這邊所看到的 **UML** 圖都是使用 **Jude** 繪製的。

## Gof 模式

以下的設計模式則是我個人從 Gof 學習中的個人體會與實作，並增加幾個導入或衍生的簡單模式。

- Creational 模式

  如何有效率的產生、管理與操作物件，一直都是值得討論的課題，Creational 模式即與物件的建立相關，在這個分類下的模式給出了一些指導原則及設計的方向。

  - **Simple Factory** 模式
  - **Abstract Factory** 模式
  - **Factory Method** 模式
  - **Builder** 模式
  - **Prototype** 模式
  - **Singleton** 模式
  - **Registry of Singleton** 模式

- Structural 模式

如何設計物件之間 的靜態結構，如何完成物件之間的繼承、實 現與依賴關係，這關乎著系統設計出來是否健壯（robust）：像是易懂、易維護、易修改、耦合度低等等議題。Structural 模式正如其名，其分類下的模式給出了在不同場合下所適用的各種物件關係結構。

- **Default Adapter** 模式
- **Adapter** 模式 **- Object Adapter**
- **Adapter** 模式 **- Class Adapter**
- **Bridge** 模式
- **Composite** 模式
- **Decorator** 模式
- **Facade** 模式
- **Flyweight** 模式
- **Proxy** 模式

- Behavioral 模式

  物件之間的合作行 爲構成了程式最終的行爲，物件之間若有設 計良好的行爲互動，不僅使得程式執行時更有效率，更可以讓物件的職責更爲清晰、整個程式的動態結構（像是物件調度）更有彈性。

  - **Chain of Responsibility** 模式
  - **Command** 模式
  - **Iterator** 模式
  - **Strategy** 模式
  - **Template Method** 模式
  - **Observer** 模式
  - **Mediator** 模式
  - **State** 模式
  - **Memento** 模式
  - **Visitor** 模式
  - **Interpreter** 模式

## 多執行緒模式

在很多應用中都會使用多執行緒，尤其是在Web應用中，多執行緒以 Gof 整理的模式爲基礎，考量多執行緒環境中，如何組合這些基本模式來完成多執行緒安全要求。

- **Guarded Suspension** 模式
- **Producer Consumer** 模式
- **Thread-Per-Message** 模式
- **Worker Thread** 模式
- **Thread Pool** 模式

- **Future** 模式
- **Read-Write-Lock** 模式
- **Two-phase Termination** 模式
- **Thread-Specific Storage** 模式

## 參考資料

以下是以Java實作設計模式的介紹網站，從下面的連結開始，當中您可以找到更多設計模式的資源。

- **Huston Design Pattern**
- **The Design Patterns Java Companion**
- 板 橋里人的 **Java** 設計模式學習心得
- **Essential JavaScript & jQuery Design Patterns**
- **UML** 軟件工程組織