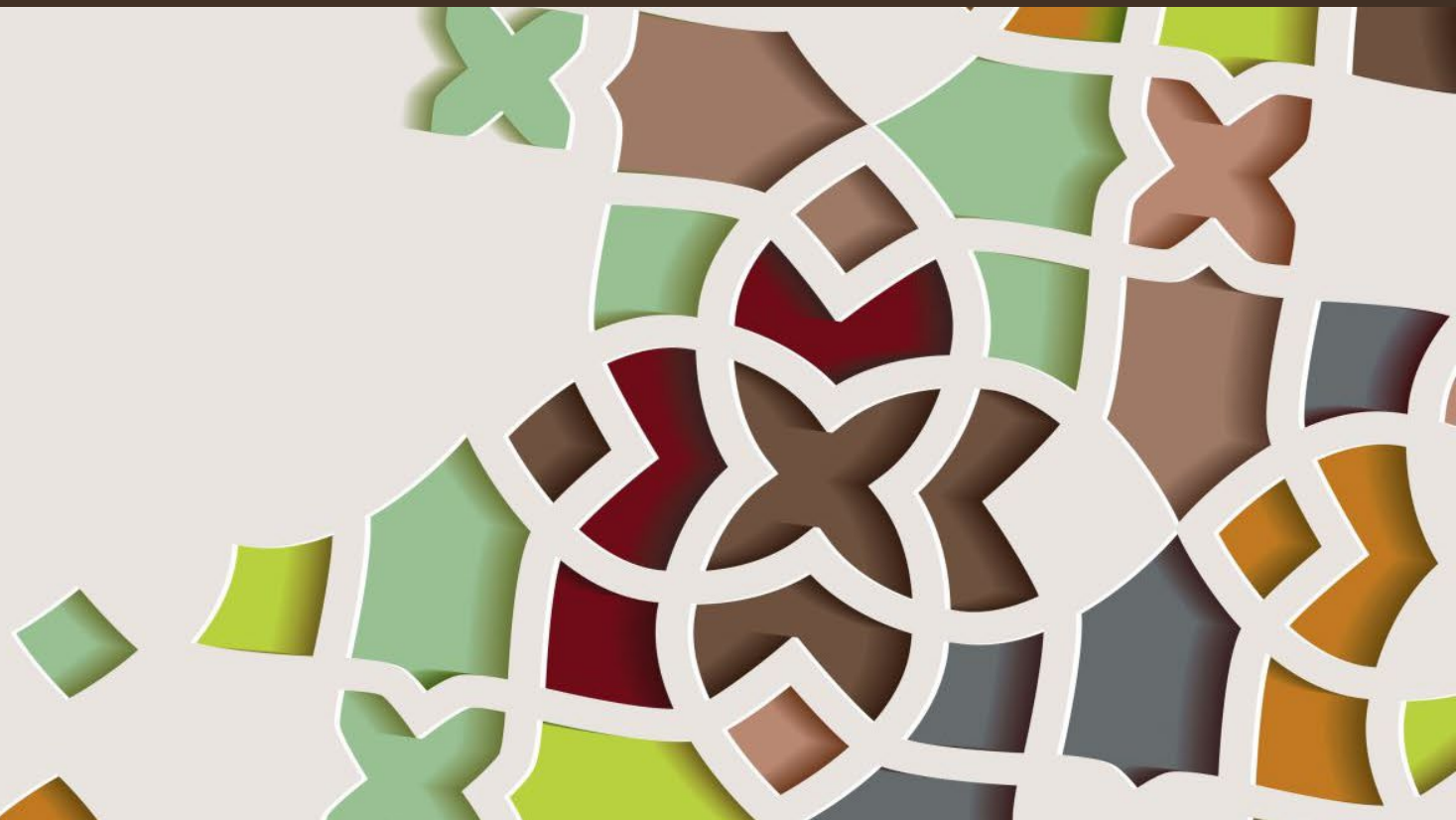
Design Pattern

Lecture 07



回顧

- → 2種模式
  - → 訪問者模式
  - → 複合模式
- → 模式分類
- → UML圖

本回

→ MVC

→ PureMVC

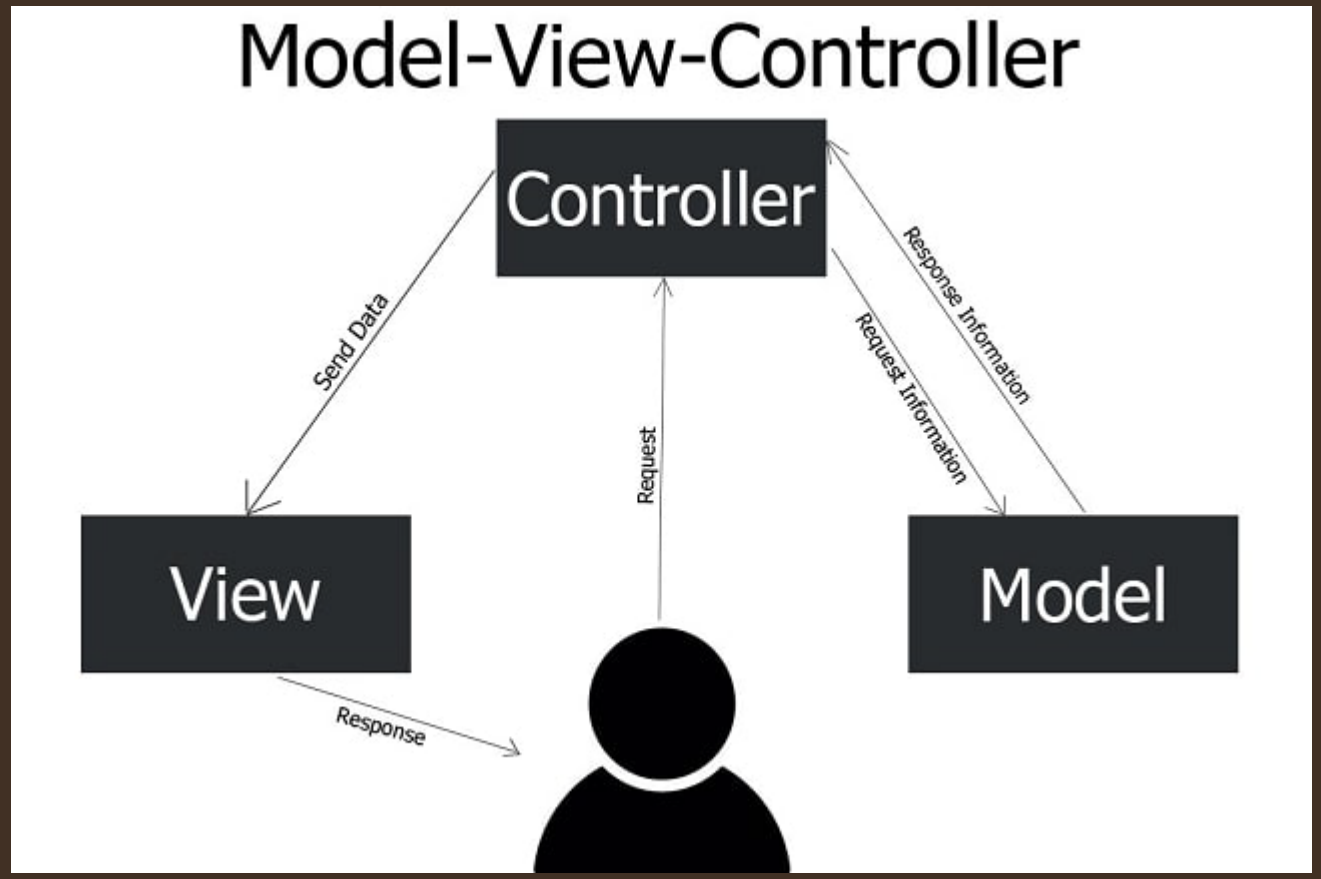
MVC

→ MVC是一種思想，它將「顯示」「數據存儲」「控制」的責任分離成不同的類處理。而實現MVC的方法有很多種。

→ M：Model（數據）

→ V：View（視圖）

→ C：Controller（控制器）



PureMVC 內容

- PureMVC是MVC的實現方式之一。

- 框架GitHub: https://github.com/PureMVC/puremvc-csharp-standard-framework
- 參考資料:
  - PureMVC框架在Unity中的應用(一) https://gameinstitute.qq.com/community/detail/127468
  - PureMVC框架在Unity中的應用(二) https://gameinstitute.qq.com/community/detail/127518
  - PureMVC框架在Unity中的應用例子 https://github.com/kenrivcn/PureMVC_Demo
  - Official Website(含中文): http://puremvc.org/

PureMVC介紹

- PureMVC在MVC之上再引入了另外的設計模式，分別是
  - Proxy
  - Mediator
  - Command
  - Façade

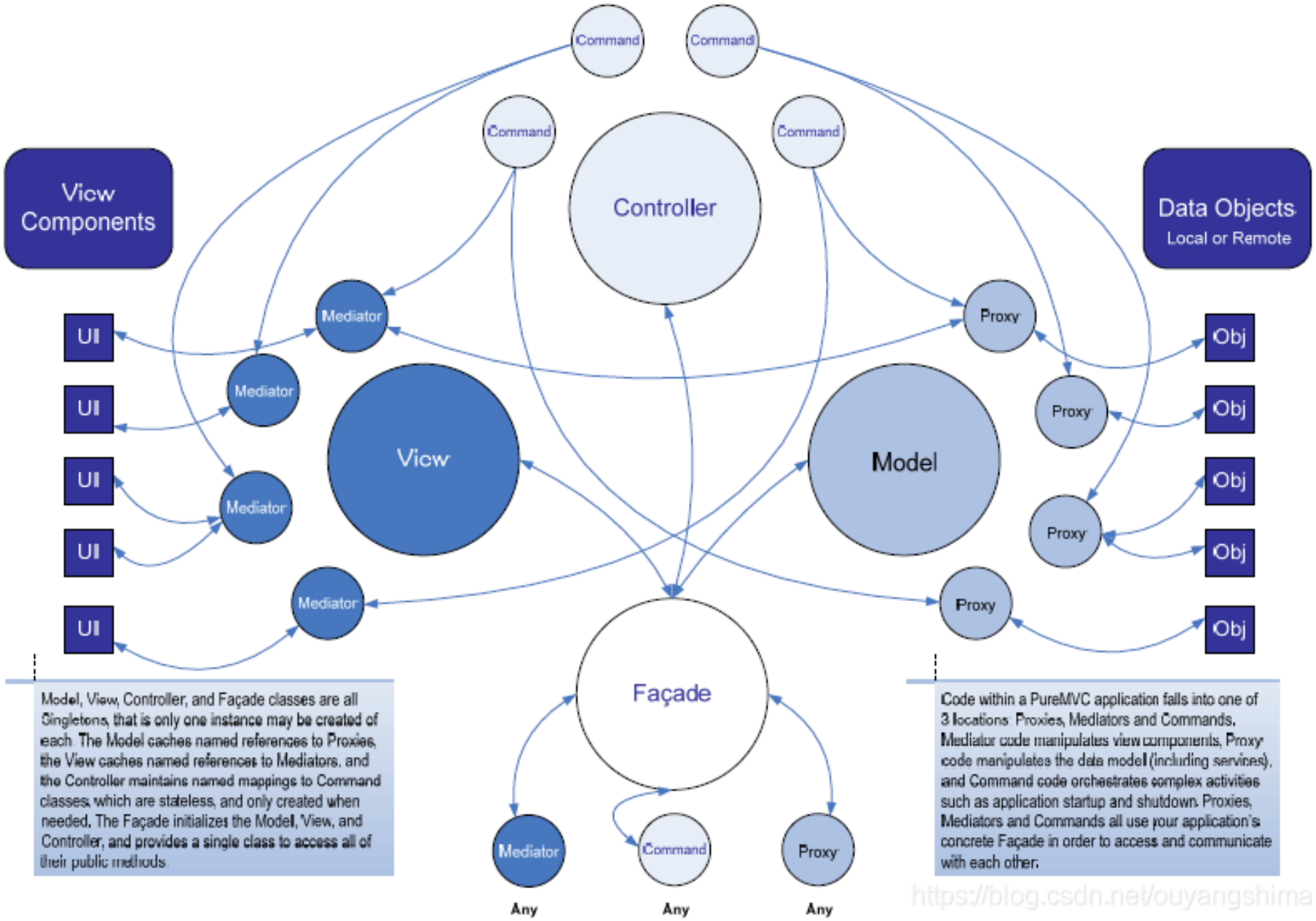
PureMVC的三大核心

- → View & Mediator
- → Model & Proxy
- → Controller & Command

- → View & Model & Controller相當於一個Manager類，提供了Register、Unregister、Retrieve等的操作。並且保存了Mediator、Proxy、Command等的字典引用。

puremvc framework

CONCEPTUAL DIAGRAM AND INTRODUCTION



PureMVC的模塊 - Facade

- → PureMVC中的Façade是包含了Controller、View、Model所有操作的一個接口集合。

- → 也是提供了一個對外的統一接口
  - → Command
  - → 其它第三方用戶

PureMVC的模塊 - Notification

- PureMVC中各系統上的交互主要通過”通知”進行的。

- 提供了三個類實現通知系統: Notifier、Notification、Observer。
  - Notifier: 通知者。提供了SendNotification
  - Notification: 事件。事件中會有名字、發送者、類型以及其內容。
  - Observer: 監聽者。用於監聽特定的Notification，並且會執行callback函數。

- Notifier中因為經常會使用到Façade，所以保存了一個對Façade單例的引用。
- Proxy就是一個Notifier，當數據改變時，會發送一個「DataChanged」的Notification。
- Mediator就是一個Observer，它會監聽自己感興趣的Notification。(Observer全部集中在View中管理)

View中的ObserverMap

Key (Notification Name)	Value (Observer)
BattleDataChanged	BattleViewObserver PlayerHPBarObserver
PlayerDataChanged	PlayerInfoUIObserver PauseUIObserver
…	…

*Notification的名字可以用const來防止一些錯誤。

PureMVC的模塊 – Proxy & Model

→ Proxy隔開了Model的直接訪問，使Model的重用性更高。例如，有多種操作可能都是使用同一個Model，這樣這種操作就是對應每一個Proxy，但Model並沒有改變。

→ Proxy是Notifier，當數據發生改變時，會執行SendNotification。

PureMVC的模塊 – Mediator & View

→ View負責管理通知系統中的Observers。

→ 每一個Mediator對應一個UI。所有對於UI的操作都被封裝到Mediator當中。而Mediator本身會對一系列的Notifications感興趣，所以提供了一個string[]來保存所有感興趣的Notification的名字，在創建時注冊到View中。

→ 另外，由於Mediator會很頻繁訪問Proxy以及UI，所以會保存Proxy和UI的引用，方便使用。

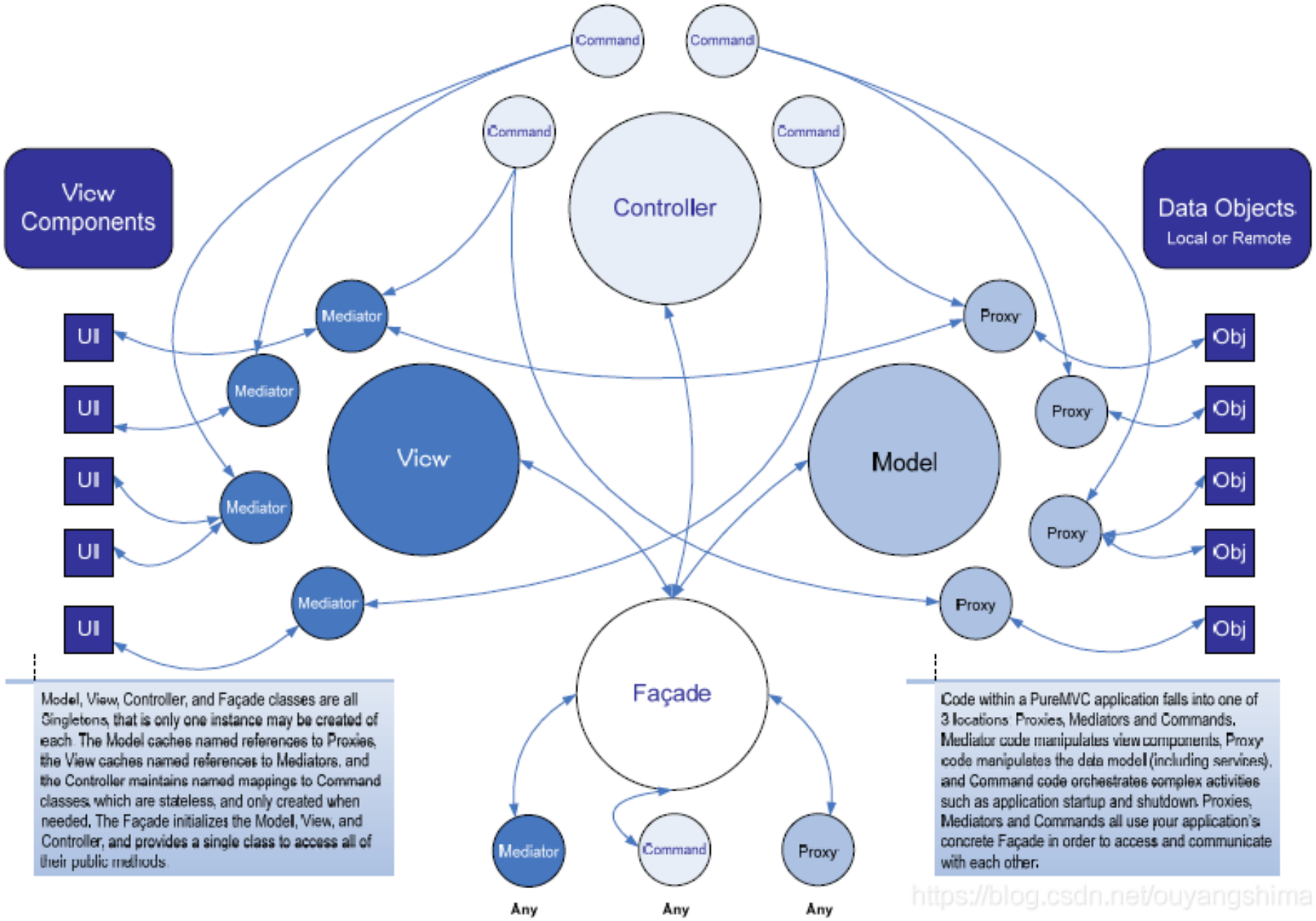
PureMVC的模塊 – Command & Controller

- 所有對應的操作都是通過Command來實現的，例如
  - 顯示UI
  - 啟動程序
  - 刷新頁面
  - …
- Command可以通過Façade訪問到Controller、View以及Model，也可以發送Notification或者執行其它的Command。

Proxy v.s. Mediator?

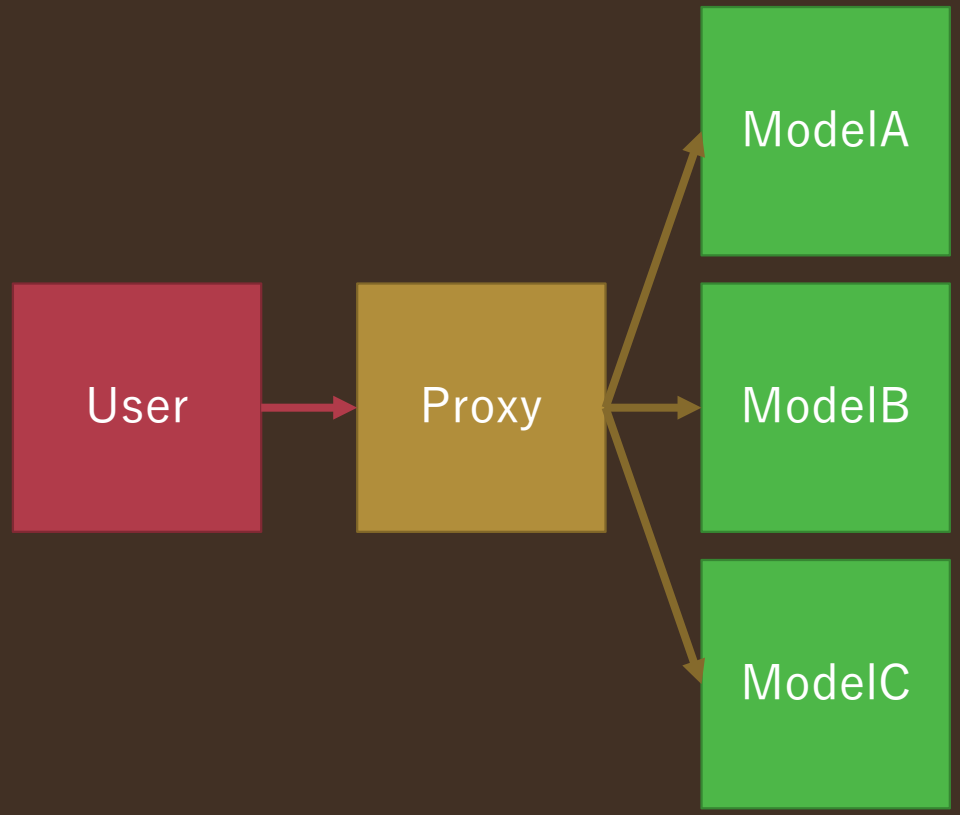
puremvc framework

CONCEPTUAL DIAGRAM AND INTRODUCTION



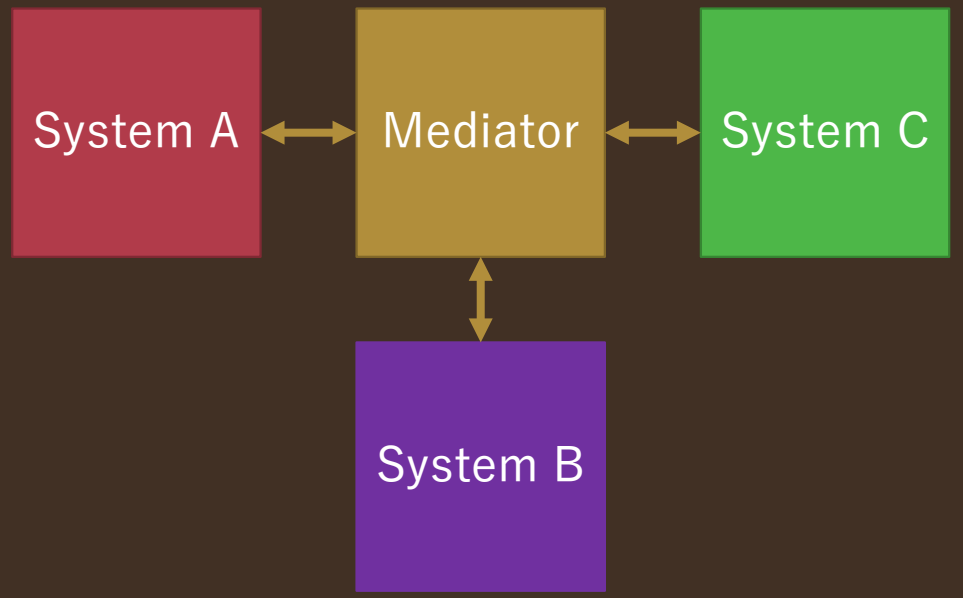
Proxy v.s. Mediator?

→ Proxy是封裝了對Model的一系列的操作的一個代理者。提供了訪問或操作Model的接口，他不會牽涉到第三者。



Proxy v.s. Mediator?

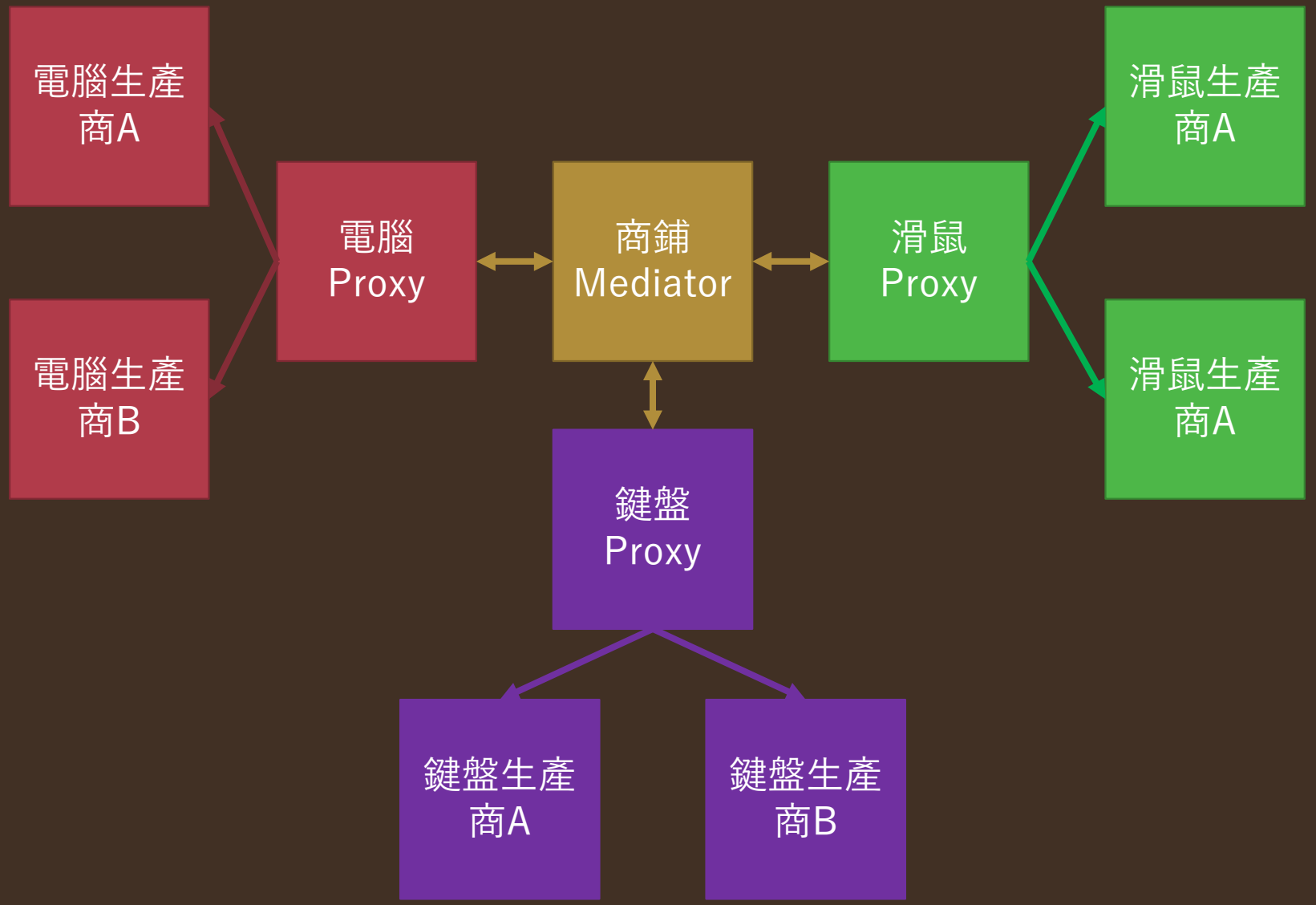
→ Mediator封裝的是對象與對象之間的交互，它不是View的代理人，而是一個中介，對象的操作可能牽涉到第三者。



Proxy v.s. Mediator?

→ 現實的例子: 有賣電腦的代理商A、他需要找賣鍵盤的代理商B、以及賣滑鼠的代理商C，ＡＢＣ之間需要交互，交互就會有一個中間人(商鋪)。

Proxy v.s. Mediator?



為甚麼Command也要利用Notification?

→ 個人認為是方便管理。

→ Controller在接受到通知後，創建Command，並執行。因此，Controller也需要Observer來監聽事件。而監聽的責任已經全部交由View來處理，所以Controller中的監聽也放到View中管理。

Q&A

Demo Time!

例子的流程

- → Application -> MyFacade.Launch
  - → Initialize Controller, View, Model
    - → 兩個Proxy、4個Command
    - → 注冊Command時會自動注冊在View中注冊Observer
- → MyFacade.SendNotification (START_UP notification)
  - → START_UP是一個Command，所以會通知Controller，並執行StartUpCommand
    - → StartUpCommand會創建MainPanelView，並將UI注冊成Mediator
      - → Mediator創建時會初始化各種東西(例如Button的onClick事件)
    - → 發送REFRESH_BONUS_ITEMS以及UPDATE_PLAYER_DATA事件
  - → REFRESH_BONUS_ITEMS是一個Command
    - → 創新列表中的12個獎品
    - → 發送REFRESH_BONUS_UI事件
  - → REFRESH_BONUS_UI和UPDATE_PLAYER_DATA由MainPanelView執行，都是用於刷新頁面

例子的流程

- → 點擊隨機獎品後，會發送PLAY事件
- → PLAY事件是一個Command
  - → 隨機獲得一個獎品
  - → 訪問PlayerDataProxy修改PlayerData
- → 當PlayerDataProxy的數據改變時會發送UPDATE_PLAYER_DATA以及REWARD_TIP_VIEW事件
  - → REWARD_TIP_VIEW事件是一個Command
    - → 他會開啟RewardTipView，並且注冊其Mediator
  - → UPDATE_PLAYER_DATA是MainPanelViewMediator負責處理的事件，會創新頁面。
- → 在RewardTipView中點擊返回後會發送REFRESH_BONUS_ITEMS，並關閉頁面
  - → REFRESH_BONUS_ITEMS是一個Command，由Controller創建Command並執行

大總結

- → 學習內容
  - → 7種原則
  - → 22種設計模式
  - → 模式分類
  - → PureMVC
- → 多用，多思考，多看

*Entity Component System(ECS)

Q&A