軟體工程作業三

架構模式(Architectural Patterns):

The architectural patterns address various issues in [software engineering](https://en.wikipedia.org/wiki/Software_engineering), such as [computer hardware](https://en.wikipedia.org/wiki/Computer_hardware) performance limitations, [high availability](https://en.wikipedia.org/wiki/High_availability) and minimization of a [business risk](https://en.wikipedia.org/wiki/Business_risk). Some architectural patterns have been implemented within [software frameworks](https://en.wikipedia.org/wiki/Software_framework).

併發模式(Concurrency Patterns):

**applications must handle multiple tasks in a manner that simulates parallelism.**

**在許多的作業系統中，往往會採用Multiprogramming的方式來進行程序的管理(process management)，而此應用便是用上了架構模式中的併發模式，此設計可以使管理更加的有效益，可在一程序進入等待時利用Time Sharing 去處理其它能處理的程序。**

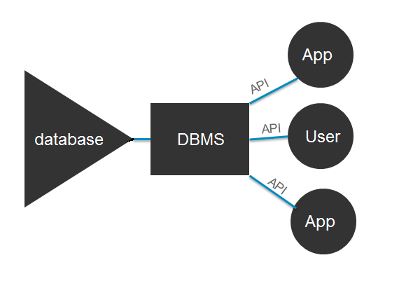
**而task scheduler，也是在併發模式的範疇內，它利用的是將欲執行的任務進行最有效率的排程，使其能在短時間內完成多的任務，提高效率。**

持續模式(PersistencePatterns):

**Data persists if it survives past the execution of the process that created it.**

**‧DBMS:**

**makes it possible for end users to create, read, update and delete**[**data**](http://searchdatamanagement.techtarget.com/definition/data)**in a database.**

****

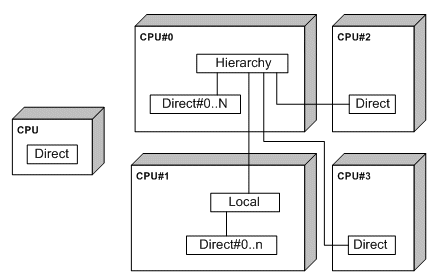
**DBMS扮演著溝通database與APP、Use間的角色，此為Persistence Patterns的其中一種應用。**

**第二種運用則是在Application Layer 的時候，分層必續持續持久，這樣才會有比較持久的應用架構。**

分散模式(DistributionPatterns):

**the manner in which systems or components within systems communicate with one another in a distributed environment**

**通常用在那些較為小，介於系統至系統間的溝通程序，通稱為brokers，他們便是分散模式的應用**

****