# Actor Model, Virtual Actor Model以及Orleans提供API的基本單元概念解釋

## Actor Model 平行運算模型概念

Actor 是一種不考慮實際實作上是Process還是Thread的抽象概念運算單元，擁有下列特性： \* Actor有自身的狀態資料，並且只有Actor自身實體有辦法修改。 \* Actor實體在整個系統的識別上/定址上是唯一的，以便讓其他Actor實體發送訊息給它。 \* Actor可接收客戶端或其他Actor所發送的訊息，不限於本機或遠端。 \* 同時刻一個Actor只能處理一則訊息，處理完了之後才能再處理下一則。 \* Actor除了根據接收之訊息種類與內容之外、也可納入目前自身狀態資料，選擇要執行的對應動作。 \* 前述所說的對應動作除了跑開發者撰寫的程式邏輯產生運算結果回傳或修改自身狀態的功能之外，也可能會建立其他子Actor實體，以便發送訊息給它驅動其開始運作。 \* 發送訊息給其他Actor的動作是*容錯*的：假如發送給某子Actor訊息並接收回傳結果的動作失敗，母Actor可嘗試改發送給另一個子Actor來代替。 \* 子Actor實體的生命週期由母Actor實體來管理：母Actor可建立新的子Actor實體以發送訊息驅使其執行運算，也可提前終止子Actor的運算執行。

從這些抽象字義上看，Actor Model與物件導向程式(OOP)所討論的類別/物件，以及物件之間發送訊息的設計概念上頗相似，但存在差異： \* Actor Model的Actor都是單獨實體，不會有OOP常見的物件繼承階層關係。 \* OOP的物件繼承階層關係，是在撰寫程式時就決定好的，而Actor Model的Actor實體的生命週期是由母Actor實體來管理，因此可以在運行時期動態決定要建立的子Actor實體種類和數量。 \* Actor Model的Actor發送的訊息是非同步的，而OOP的物件發送的訊息是同步的。 \* Actor Model不管實際上Actor實體們是位於同一台機器上，還是分散在不同機器上，發送訊息的方式都是一致的。

Actor Model的概念，這如果光看上述文字說明，其實還是很難理解，接下來會以一個我們實際生活上可能會碰到的的情境範例來比喻說明。

### Actor Model的生活情境範例：農曆過年大掃除

除夕下午三點半，在遠方的阿公打電話給媽媽說晚上會來吃年夜飯過年並住上兩三天，為了避免長輩看到家裡平日不修邊幅東西亂丟的『亂室佳人』景象，我們全家要趕快先把家裡的東西都收拾好，於是媽媽開始發號施令，叫我們三兄弟各自盡快把自己房間東西收拾好，叫老爸趕快把陽台衣服收進來，又叫他趕快把客廳亂放雜誌收一收然後掃地拖地，以便阿公來的時候，家裡的狀況是整潔的。

以上就是一個生活情境比喻的Actor Model運算架構範例，解釋如下：

* 這個Actor Model的客戶端是阿公，發送訊息(打電話)給媽媽這個母Actor實體之後，然後媽媽就開始『一一指令』(發訊息)給幾個子Actor實體：我們三兄弟＆老爸。
* 三兄弟＆老爸是媽媽的子Actor實體，他們收到媽媽的『指令』後，就開始執行『對應動作』(處理訊息)：三兄弟開始各自收拾自己房間的東西，老爸開始收陽台衣服。
* 三兄弟各自收拾房間，自己的房間只有自己會收拾，不會去收拾別人的房間：Actor有自身的狀態，也只有Actor自身有權現處理自己的狀態。
* 三兄弟收拾房間的動作，是非同步的，也就是說，三兄弟收拾房間的動作，不會互相干擾，也不會互相等待，而是各自收拾自己的房間，直到收拾完畢，才會發送訊息給媽媽，告訴媽媽自己的房間收拾完畢了。
* 老爸把陽台衣服收進來之後，又開始打掃客廳：一個Actor實體一次只能處理一個訊息，而且要等到這個訊息處理完畢，才會繼續處理下一個訊息。

當然，這個例子只是為了讓大家更容易理解Actor Model的概念，沒有對應到Actor Model一個很重要的特性：

**隨著負載數量級的增加，Actor Model的架構能輕鬆地水平擴展(scale-out)**，也就是說，當客戶端傳送的訊息數量逐步增加時，Actor Model架構也可增加處理訊息的Actor實體，以便在整體系統的『吞吐量(throughput)』上保持穩定的處理速度，避免客戶端數量越多系統就越慢的問題；而由於Actor在發送訊息的方法在概念上不分本地端/遠端都是一樣的，所以Actor Model架構能在多台機器上部署Actor實體而不需要修改撰寫的程式碼，並且在整體系統的『可用性(availability)』上，避免系統因為某台獨特的關鍵伺服器故障而造成整體系統不可用。

以上是Actor Model的說明，如果想要看Actor Model發明者對於Actor Model的實質討論，可以看這篇[The actor model in 10 minutes](https://www.brianstorti.com/the-actor-model/)的Q&A影片。  
以及這篇[Down and Dirty: Understanding the Actor Model](https://www.developer.com/design/down-and-dirty-understanding-the-actor-model/)的範例解說。

## Virtual Actor Model 簡介

Virtual Actor Model一句話來說，就是 **不用管理Actor實體生命週期的Actor Model** ：只要有目標Actor實體的識別子(Identifier)，就能對該Actor實體發送訊息叫它工作，不需控管子Actor實體的實際生命週期，這些瑣碎事務由實作框架的內部運作機制就幫開發者處理掉，減少開發時的思考負擔。  
(有種程式語言發明了Garbage Collection機制後就不必再去思考記憶體管理的感覺)

## Orleans提供API基本單元概念

Orleans框架提供的API基本單元，對應到Virtual Actor Model的，有以下幾個：

### Grain

也就是Actor Model的Actor。

### Silo

容納Grain的容器，一個Silo可以容納多個Grain，一個Grain只能屬於一個Silo，有點類似K8s中Pod對應於container的關係。

### Cluster

實際跑Grain運算的實體/虛擬機，在Orleans的API中通常以 SiloHost 來稱呼，但由於現在容器化部署方式流行之後，幾乎都是一個Silo就一個SiloHost對應的架構來部署，所以後來Orleans的運營(ops)相關API也都將對SiloHost為配置標的之部分逐步刪減，改成直接對 Silo物件為配置標的。

### Client

也就是Actor Model中的訊息起始客戶端。

對於Grains的詳細組成結構和物件宣告的實際程式碼寫法，明天繼續介紹(終於要捲起袖子開幹了)。