# Orleans Grain RPC呼叫的三事：Request Context, One-way request, Stateless Worker

Orleans在開發(dev)Grain Method相關還有一些特殊的API，這篇文章就來介紹一下：**Request Context**, **One-way request**, **Stateless Worker** 的用法。

## Request Context

Request Context 類似 [ASP.NET MVC的 ViewData Dictionary](https://learn.microsoft.com/aspnet/core/mvc/views/overview?view=aspnetcore-6.0#viewdata)，提供一個穿越RPC呼叫方法的資料傳遞管道，可以在呼叫RPC方法之前設定資料，然後在運算進入呼叫方法實作的時候，就可以取得這些資料。

Request Context的設定方法為呼叫其靜態方法 [RequestContext.Set()](https://learn.microsoft.com/dotnet/api/orleans.runtime.requestcontext.set)，此呼叫可在Grain Method實作中進行，也可以在一開始要啟始RPC方法呼叫前的Client端執行。

而讀取資料的方法為呼叫其靜態方法 [RequestContext.Get()](https://learn.microsoft.com/dotnet/api/orleans.runtime.requestcontext.get)，此呼叫可在Call Filter和Grain Method實作中進行。

Request Context 可用來設定分散式系統常需要的TraceID，以便在記錄呼叫的Log時，將TraceID一併記錄下來，方便後續的追蹤。

## One-way request

有些Grain的RPC方法如果只是要Grain去做事，不需要回傳結果，那就可以使用加掛 [[OneWay]](https://learn.microsoft.com/dotnet/api/orleans.concurrency.onewayattribute) 屬性的方式來定義Grain Method：

using Orleans;  
using Orleans.Concurrency;  
  
public interface IDemoOneWayGrain : IGrainWithGuidKey  
{  
 [OneWay]  
 Task LogSomething(string message);  
}

One-way Request的方法，回傳值只能為Task或ValueTask，不能是有回傳值的泛型 Task<T> / ValueTask<T>。

One-way Request的方法假如拋出例外，不會回傳至呼叫端。但其Incoming & Outgoing Call Filter還是會被執行。

## Stateless Worker

Stateless Worker Grain 是在Grain實作類別上加掛 [[StatelessWorker]](https://learn.microsoft.com/dotnet/api/orleans.concurrency.statelessworkerattribute) 屬性的方式來定義Grain：

using Orleans;  
using Orleans.Concurrency;  
  
[StatelessWorker]  
public class MyWorker : Grain, ...  
{  
 // Grain implementation   
}

此種實作的Grain，有下列特點：

1. 如果是以Grain-to-Grain的呼叫其RPC方法時，會將此Grain啟用在和呼叫者同一台的Silo上，以達最佳的效能。
2. 此種Grain的定義識別子(Grain Identity)，沒有太大意義，因為Orleans Runtime會在每次呼叫該種Grain的RPC方法時，都會重新啟用一個新的Grain實例，以便讓RPC呼叫的反應速度達到最快。
3. 承接1和2，假如該Grain的RPC呼叫需求很頻繁時，Orleans Runtime 會在每台Silo上啟用多個Grain實例，以達到最佳的效能。（在每台Silo上啟用的最大數量限制可由 [StatelessWorker] 屬性的另一個有整數值輸入值的建構式來控制）

此種Grain適合拿來執行一些不會改變Grain狀態、需要可以很快速產生輸出的運算類型，也能拿來當作某種類似『Process Pool』的用途。

此種Grain雖然也可以使用DI倚賴注入狀態變數，但不建議這樣做，因為每次呼叫RPC方法時，都會啟用一個新的Grain實例，所以有可能狀態變數會同時被多個Grain實例改變。建議的作法是，對於要有狀態修改的動作，使用設計模式(Design Pattern)之一的委派模式(Delegate Pattern)，將需要改變的資料，存放於另一個正常Grain實例的狀態之中，然後Stateless Grain去呼叫該正常Grain所開放的修改狀態RPC方法之方式來變更。

明天來介紹如何套用Polly的Retry Policy來處理當Grain的RPC呼叫時的連線狀態不穩定問題，以及一些Silo配置程式碼的注意事項。