# Orleans Grain的Timer和Reminder機制

Orleans 提供Grain兩種方式來實現定時任務，一種是 Timer，另一種是 Reminder。

Timer不需要在Silo端額外配置紀錄設定用的Provider，而Reminder需要，Orleans框架有提供內建 In-Memory的 Provider，以方便測試之用，官方也有提供將Reminder設定儲存至Azure Table Storage，以及 ADO.NET連線的SQL資料庫的Provider，藉由安裝官方Nuget套件就可以使用。

## Grain的Timer使用法

使用Timer的方法是在Grain實作專案的程式碼內呼叫 Grain基礎類別提供的 [RegisterTimer](https://learn.microsoft.com/dotnet/api/orleans.grain.registertimer) 函式，如以下使用Timer每秒鐘印一筆Log的Grain程式範例：

public class TimerProducerGrain : Grain, IProducerGrain  
{  
 private readonly ILogger<TimerProducerGrain> \_logger;  
  
 private IDisposable? \_timer;  
  
 private int \_counter = 0;  
   
 public TimerProducerGrain(ILogger<TimerProducerGrain> logger )  
 {  
 \_logger = logger;  
 }  
   
 public Task StartProducing()  
 {  
 if (\_timer is not null)  
 {  
 throw new Exception("This grain is already producing events");  
 }  
   
 //Register a timer that produce an event every second  
 var period = TimeSpan.FromSeconds(1);  
 \_timer = RegisterTimer(TimerTick, null, period, period);  
   
 \_logger.LogInformation("I will produce a new event every {Period}", period);  
 return Task.CompletedTask;  
 }  
  
 private Task TimerTick(object \_)  
 {  
 var value = \_counter++;  
 \_logger.LogInformation("Producing event {EventNumber}", value);  
 }  
  
 public Task StopProducing()  
 {  
 if(\_timer is not null)  
 {  
 \_timer.Dispose();  
 \_timer = null;  
 }  
 return Task.CompletedTask;  
 }  
}

Grain Timer API提供的創建語法上和一般 *.NET BCL(Base Class Library)* 提供的 [System.Threading.Timer](https://learn.microsoft.com/dotnet/api/system.threading.timer) 很像，第一個參數是Timer觸發時會執行的 Func<Object, Task> 型態的開發者自行撰寫delegate，第二個參數是呼叫該delegate時，會輸入的參數，第三個參數是Timer的第一次觸發間隔時間，第四個參數是Timer的重複觸發間隔時間，如果第四個參數是 TimeSpan.Zero，則Timer只會觸發一次。

Grain的Timer要停止，必須呼叫其執行 RegisterTimer() 方法回傳的 IDisposable 物件的 Dispose() 方法，並且也沒有提供 Change() 方法來修改或停止Timer。

Grain Timer適合使用於觸發間隔時間小於一分鐘的事務，如果是需要間隔時間大於一分鐘，並且跨Grain的起始→閒置→休眠的生命週期之事務，建議使用Reminder。

## Grain的Reminder使用法

Reminder是需要有在Silo層設定Provider才能使用的功能，但它可以跨Grain的生命週期，即使在Reminder觸發時Grain已經是從記憶體中移除的休眠狀態，Orleans Runtime也會讓該Grain重新啟動，並且執行Reminder的API所對應開發者撰寫的事件處理函式實作部分。

### Reminder寫法

要使用Reminder的Grain類別定義，必須實作 [IRemindable](https://learn.microsoft.com/dotnet/api/orleans.iremindable) 介面提供的 [ReceiveReminder()](https://learn.microsoft.com/dotnet/api/orleans.iremindable.receivereminder) 事件處理函式，這是讓Reminder觸發時執行的程式，而一開始需要在某些進入點呼叫 [RegisterOrUpdateReminder()](https://learn.microsoft.com/dotnet/api/orleans.grain.registerorupdatereminder) 方法來註冊Reminder，如此以便讓Reminder正常排程觸發：

public class MyReminderGrain : Orleans.Grain, IMyReminder, IRemindable  
{  
 private readonly ILogger<MyReminderGrain> \_logger;  
 private readonly Dictionary<string, ReminderInfo> \_registeredReminders = new ();  
  
 public MyReminderGrain(ILogger<MyReminderGrain> logger)  
 {  
 \_logger = logger;  
 }  
  
 // call this method to register a reminder  
 public async Task RegisterReminder(string reminderName, int upperLimit)  
 {  
 if(\_registeredReminders.ContainsKey(reminderName))  
 {  
 throw new Exception($"Reminder {reminderName} is already registered");  
 }  
 var reminder = await RegisterOrUpdateReminder(reminderName, TimeSpan.FromSeconds(30), TimeSpan.FromMinutes(1));  
 \_registeredReminders[reminderName] = new ReminderInfo { Reminder = reminder , UpperLimit = upperLimit };  
  
  
 return Task.FromCompleted();  
 }  
  
 /\*   
 other RPC methods ...  
 \*/  
  
 public async Task ReceiveReminder(string reminderName, TickStatus status)  
 {  
 \_logger.LogInformation("Reminder {ReminderName} is triggered", reminderName);  
  
 var reminderInfo = \_registeredReminders[reminderName];  
  
 reminderInfo.CalledCount++;  
  
 if (reminderInfo.CalledCount >= reminderInfo.UpperLimit)  
 {  
 await UnregisterReminder(reminderInfo.Reminder);  
 \_outputMsg.Output($"Reminder: {reminderName} unregistered.");  
 \_registeredReminders.Remove(reminderName);  
 }  
 }  
}  
  
//Custom class to hold reminder info  
public class ReminderInfo  
{  
 public int CalledCount { get; set; } = 0;  
 public int UpperLimit { get; set; } = 10;  
  
 public IGrainReminder Reminder { get; set; }  
}

Reminder在不需要時，可以呼叫 [UnregisterReminder()](https://learn.microsoft.com/dotnet/api/orleans.grain.unregisterreminder) 方法來取消註冊，但因為此方法需要一個 [IGrainReminder](https://learn.microsoft.com/dotnet/api/orleans.runtime.igrainreminder) 輸入參數，因此在Grain本身需準備儲存已註冊Reminder的成員變數，此範例是用一個叫 \_registeredReminders 的Dictionary資料結構變數來存。

### Reminder的Provider安裝與Silo設定

內建的In-Memory Reminder Provider設定方法如下，需呼叫 [UseInMemoryReminderService()](https://learn.microsoft.com/dotnet/api/orleans.hosting.silohostbuilderreminderextensions.useinmemoryreminderservice) 擴充方法：

.UseOrleans(builder =>  
{  
 builder.UseInMemoryReminderService();  
 /\*  
 other configuration  
 \*/  
})

官方提供的Azure Table Storage Reminder Provider設定方法如下，需安裝 [Microsoft.Orleans.Reminders.AzureStorage](https://www.nuget.org/packages/Microsoft.Orleans.Reminders.AzureStorage) 套件以便呼叫 [UseAzureTableReminderService()](https://learn.microsoft.com/dotnet/api/orleans.hosting.silohostbuilderreminderextensions.useazuretablereminderservice) 擴充方法：

.UseOrleans(builder =>  
{  
 builder.UseAzureTableReminderService(  
 options => options.ConnectionString = "The\_Azure\_Table\_Conn" );  
 /\*  
 other configuration  
 \*/  
})

官方提供的ADO.NET Reminder Provider設定方法如下，類似使用 ADO.NET 的Grain Storage Provider一樣，要在目標資料庫建立資料表，而[建立資料表的SQL Script](https://learn.microsoft.com/dotnet/orleans/host/configuration-guide/adonet-configuration#persistence)要依使用的資料庫來選擇要執行的種類下載執行；在Silo配置方面，需安裝 [Microsoft.Orleans.Reminders.AdoNet](https://www.nuget.org/packages/Microsoft.Orleans.Reminders.AdoNet) 套件以便呼叫 [UseAdoNetReminderService()](https://learn.microsoft.com/dotnet/api/orleans.hosting.silohostbuilderreminderextensions.useadonetreminderservice) 擴充方法：

.UseOrleans(builder =>  
 {  
 builder.UseAdoNetReminderService(  
 options => {  
 options.ConnectionString = "The\_AdoNet\_Conn";  
 options.Invariant = "The\_matched\_AdoNet\_Invariant"; );  
  
 /\*  
 other configuration  
 \*/  
 })

明天繼續介紹Orleans Grain的另一個重要功能：Grain Observer，事件(event)觸發功能。