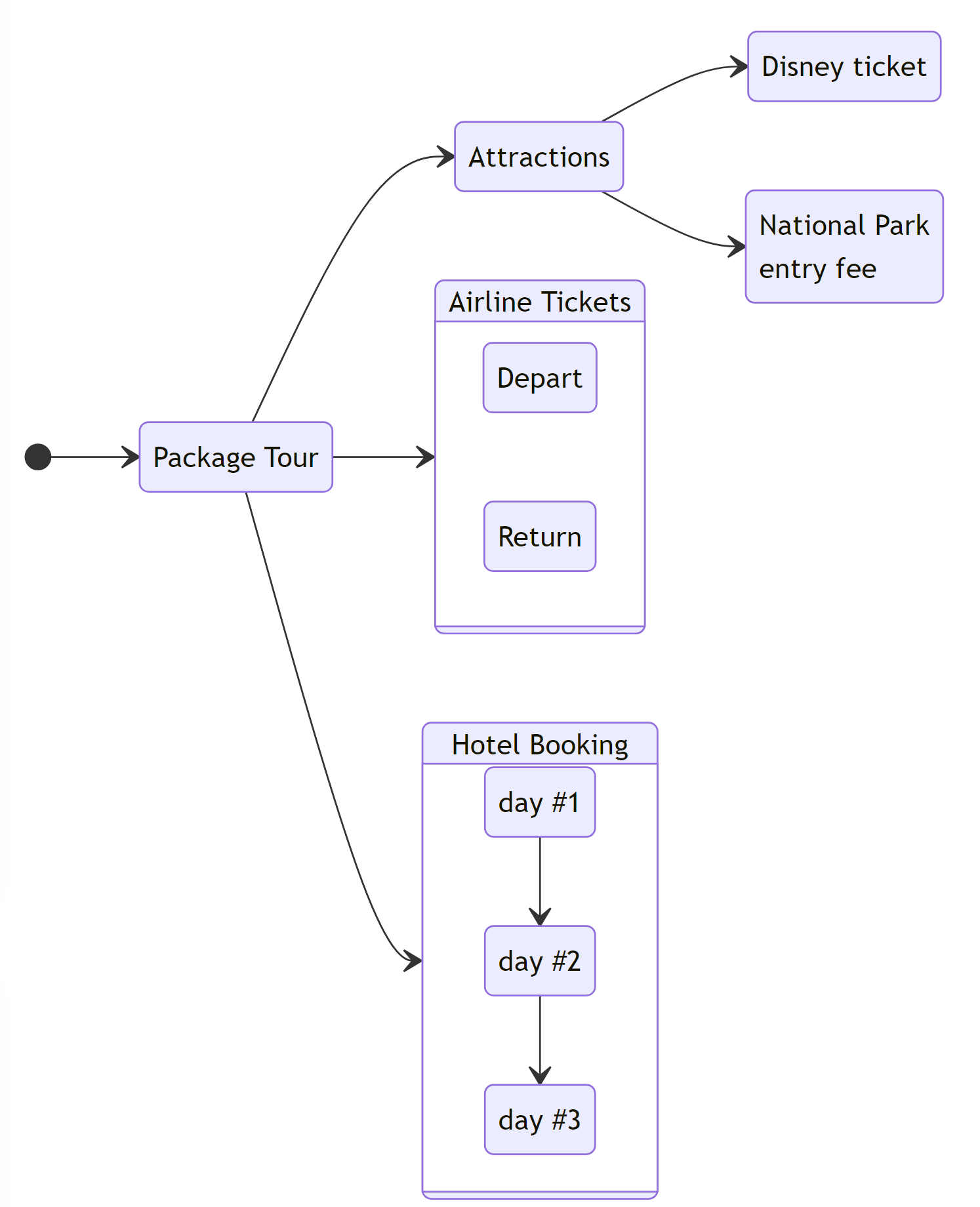
# Orleans Grain的巢狀Transaction範例實作與單元測試

## Orleans巢狀Transaction範例

昨天示範基本的Orleans Grain RPC Transaction功能，不過Orleans的Transaction是可以巢狀(nested)呼叫的，今天來示範一個使用巢狀Transaction的範例：

旅行社賣所謂的『機加酒』包裝行程，包含飛機票、住宿、早餐、午餐、晚餐、交通、導遊等等，這些每一個都是機加酒全包行程的必要元素，而這些元素都是由不同的人或商業單位來處理，一定要全部都配合好，否則其中一項預約安排不到，這個包裝行程就等於整個無效。

使用Orleans的Transaction功能來實作這個nested Transaction的Grain RPC呼叫範例，假設在商業邏輯上有如下圖的層次關係：



以下我們來用Orleans的nested Transaction來實作這個多層次的巢狀關係

### 旅行社機加酒行程範例實作

1. 使用昨天建立好的Github專案，新增下列 .NET 專案：

| * 路徑 | * 專案名稱 | * 專案類型 | * 目標框架 |
| --- | --- | --- | --- |
| * src/Shared | * **TravelPackageTour.Interfaces** | * 類別庫(class library) | * .NET 6.0 |
| * src/Grains | * **TravelPackageTour.Grains** | * 類別庫(class library) | * .NET 6.0 |
| * src/Hosting/Client | * **TravelPackageTour.Client** | * 主控台應用程式(Console App) | * .NET 6.0 |
| * src/Hosting/Server | * **TravelPackageTour.Silo** | * Worker Service | * .NET 6.0 |

* 將這些專案各自加入根目錄 OrleansTransactionDemo.sln 方案的方案資料夾(Solution Folder)中。

1. 設定專案參考：
   * 將 TravelPackageTour.Interfaces 專案加入到 TravelPackageTour.Client 專案的專案對專案參考(Project reference)中。
   * 將 TravelPackageTour.Interfaces 專案加入到 TravelPackageTour.Grains 專案的專案對專案參考(Project reference)中。
   * 將 TravelPackageTour.Grains 專案加入到 TravelPackageTour.Silo 專案的專案對專案參考(Project reference)中。
2. 安裝各專案的Nuget套件：

| * 專案名稱 | * 套件名稱 | * 套件版本 |
| --- | --- | --- |
| * **TravelPackageTour.Interfaces** | * + [Microsoft.Orleans.Core.Abstractions](https://www.nuget.org/packages/Microsoft.Orleans.Core.Abstractions)   + [Microsoft.Orleans.CodeGenerator.MSBuild](https://www.nuget.org/packages/Microsoft.Orleans.CodeGenerator.MSBuild) | * 3.6.5 3.6.5 |
| * **TravelPackageTour.Grains** | * + [Microsoft.Orleans.Core](https://www.nuget.org/packages/Microsoft.Orleans.Core)   + [Microsoft.Orleans.CodeGenerator.MSBuild](https://www.nuget.org/packages/Microsoft.Orleans.CodeGenerator.MSBuild)   + [Microsoft.Orleans.Transactions](https://www.nuget.org/packages/Microsoft.Orleans.Transactions) | * 3.6.5 3.6.5 3.6.5 |
| * **TravelPackageTour.Client** | * + [Microsoft.Orleans.Client](https://www.nuget.org/packages/Microsoft.Orleans.Client)   + [Serilog](https://www.nuget.org/packages/Serilog)   + [Serilog.Extensions.Hosting](https://www.nuget.org/packages/Serilog.Extensions.Hosting)   + [Serilog.Sinks.Console](https://www.nuget.org/packages/Serilog.Sinks.Console)   + [Serilog.Sinks.Debug](https://www.nuget.org/packages/Serilog.Sinks.Debug) | * 3.6.5 2.12.0 5.0.1 4.1.0 2.0.0 |
| * **TravelPackageTour.Silo** | * + [Microsoft.Extensions.Hosting](https://www.nuget.org/packages/Microsoft.Extensions.Hosting)   + [Microsoft.Orleans.Server](https://www.nuget.org/packages/Microsoft.Orleans.Server)   + [Serilog](https://www.nuget.org/packages/Serilog)   + [Serilog.Extensions.Hosting](https://www.nuget.org/packages/Serilog.Extensions.Hosting)   + [Serilog.Sinks.Console](https://www.nuget.org/packages/Serilog.Sinks.Console)   + [Serilog.Sinks.Debug](https://www.nuget.org/packages/Serilog.Sinks.Debug) | * 6.0.1 3.6.5 2.12.0 5.0.1 4.1.0 2.0.0 |

1. 撰寫 **TravelPackageDemo.Interfaces** 專案內的程式碼，移除預設產生的 *Class1.cs* 檔案，新增下列程式碼：

* <<**IPackageTourGrain.cs**>>
* using Orleans;  
    
  namespace TravelPackageTour.Interfaces;  
    
  public interface IPackageTourGrain : IGrainWithGuidKey  
  {  
   [Transaction(TransactionOption.Create)]  
   Task BuyPackageTour();  
  }
* <<**IAttractionsGrain.cs**>> ```csharp using Orleans;
* namespace TravelPackageTour.Interfaces;
* public interface IAttractionsGrain : IGrainWithGuidCompoundKey { [Transaction(TransactionOption.Create)] Task OrderDisneyTicket();
* [Transaction(TransactionOption.Create)]  
   Task PayNationalParkEntryFee();
* } ```
* <<**IDisneyTicketBoothGrain.cs**>>
* using Orleans;  
    
  namespace TravelPackageTour.Interfaces;  
    
  public interface IDisneyTicketBoothGrain : IGrainWithIntegerKey  
  {  
   [Transaction(TransactionOption.Join)]  
   Task<DisneyTicket> GetDisneyTicket();  
  }  
    
  public record DisneyTicket(Guid ticketId, string ticketName);
* <<**INationalParkOfficeGrain.cs**>>
* using Orleans;  
    
  namespace TravelPackageTour.Interfaces;  
    
  public interface INationalParkOfficeGrain : IGrainWithStringKey  
  {  
   [Transaction(TransactionOption.Join)]  
   Task<ParkEntry> PayParkEntryFee(int amount);  
  }  
    
  public record ParkEntry(string EntryId);
* <<**IAirlineTicketAgencyGrain.cs**>>
* using Orleans;  
    
  namespace TravelPackageTour.Interfaces;  
    
  public interface IAirlineTicketAgencyGrain : IGrainWithStringKey  
  {  
   [Transaction(TransactionOption.CreateOrJoin)]  
   Task<Flight> BuyDepartAirlineTicket();  
    
   [Transaction(TransactionOption.CreateOrJoin)]  
   Task<Flight> BuyReturnAirlineTicket();  
    
   [Transaction(TransactionOption.Supported)]  
   Task<string> GetCurrentOrderStatus();  
  }  
    
  public record Flight()  
  {  
   public string Airline { get; init; }  
   public string FlightNumber { get; init; }  
   public string DepartureAirport { get; init; }  
   public string ArrivalAirport { get; init; }  
   public DateTime DepartureTime { get; init; }  
   public DateTime ArrivalTime { get; init; }  
  };
* <<**IHotelBookingGrain.cs**>>
* using Orleans;  
    
  namespace TravelPackageTour.Interfaces;  
    
  public interface IHotelBookingGrain  
  {  
   [Transaction(TransactionOption.CreateOrJoin)]  
   Task<bool> BookingHotelDayOne();  
    
   [Transaction(TransactionOption.CreateOrJoin)]  
   Task<bool> BookingHotelDayTwo();  
    
   [Transaction(TransactionOption.CreateOrJoin)]  
   Task<bool> BookingHotelDayThree();  
  }
* <<**TicketSoldOutException.cs**>>
* namespace TravelPackageTour.Interfaces;  
    
  public class TicketSoldOutException : Exception  
  {  
   public string TicketType { get; }  
    
   public TicketSoldOutException(string ticketType) : base($"Ticket type {ticketType} is sold out")  
   {  
   TicketType = ticketType;  
   }  
  }

1. 撰寫 **TravelPackageDemo.Grains** 專案內的程式碼，移除預設產生的 *Class1.cs* 檔案，新增下列程式碼：

* <<**Usings.cs**>>
* global using Microsoft.Extensions.Logging;  
  global using Orleans;  
  global using Orleans.Transactions.Abstractions;  
  global using TravelPackageTour.Interfaces;
* <<**AirlineTicketAgencyGrain.cs**>>
* using System.Transactions;  
    
  namespace TravelPackageTour.Grains;  
    
  public class AirlineTicketAgencyGrain : Grain, IAirlineTicketAgencyGrain  
  {  
   private readonly ITransactionalState<AirlineTicketStatus> \_airlineTicketStatus;  
   private readonly ILogger<AirlineTicketAgencyGrain> \_logger;  
    
   public AirlineTicketAgencyGrain(  
   [TransactionalState("airlineTicketStatus")]  
   ITransactionalState<AirlineTicketStatus> airlineTicketStatus,   
   ILogger<AirlineTicketAgencyGrain> logger)  
   {  
   \_airlineTicketStatus = airlineTicketStatus;  
   \_logger = logger;  
   }  
    
   public Task<Flight> BuyDepartAirlineTicket()  
   {  
   // Fake buy airline ticket logic, may be 50% chance to buy successfully  
   var random = new Random();  
   if (random.Next(0, 100) > 50)  
   {  
   throw new TransactionAbortedException("Airline is not available");  
   }  
    
   var flight = new Flight  
   {  
   Airline = "Airline 1",  
   FlightNumber = "Flight 1",  
   DepartureTime = new DateTime(),  
   ArrivalTime = DateTime.Now.Add(new TimeSpan(1, 0, 0)),  
   DepartureAirport = "Airport 1",  
   ArrivalAirport = "Airport 2"  
   };  
   \_airlineTicketStatus.PerformUpdate(status => status.DepartFlight = flight);  
    
   \_logger.LogInformation("BuyDepartAirlineTicket successfully");  
   return Task.FromResult(flight);  
   }  
    
   public Task<Flight> BuyReturnAirlineTicket()  
   {  
   // Fake buy airline ticket logic, may be 50% chance to buy successfully  
   var random = new Random();  
   if (random.Next(0, 100) > 50)  
   {  
   throw new TransactionAbortedException("Airline is not available");  
   }  
    
   var flight = new Flight  
   {  
   Airline = "Airline 1",  
   FlightNumber = "Flight 2",  
   DepartureTime = DateTime.Now.Add(new TimeSpan(2, 0, 0)),  
   ArrivalTime = DateTime.Now.Add(new TimeSpan(3, 0, 0)),  
   DepartureAirport = "Airport 1",  
   ArrivalAirport = "Airport 2"  
   };  
   \_airlineTicketStatus.PerformUpdate(status => status.ReturnFlight = flight);  
    
   \_logger.LogInformation("BuyReturnAirlineTicket successfully");  
   return Task.FromResult(flight);  
   }  
    
   public Task<string> GetCurrentOrderStatus()  
   {  
   return \_airlineTicketStatus.PerformRead(status =>  
   {  
   var str = $"Depart: {status.DepartFlight}, Return: {status.ReturnFlight}";  
   return str;  
   });  
   }  
  }  
    
  public record class AirlineTicketStatus()  
  {  
   public Flight? DepartFlight { get; set; }  
   public Flight? ReturnFlight { get; set; }  
  }

## Orleans Transaction單元測試

Orleans的Transaction要寫單元測試驗證時，官方提供一個Nuget套件：[**Microsoft.Orleans.Transactions.TestKit.xUnit**](https://www.nuget.org/packages/Microsoft.Orleans.Transactions.TestKit.xUnit)，可用來輔助使用xUnit的測試框架時撰寫單元測試，以下介紹如何使用。