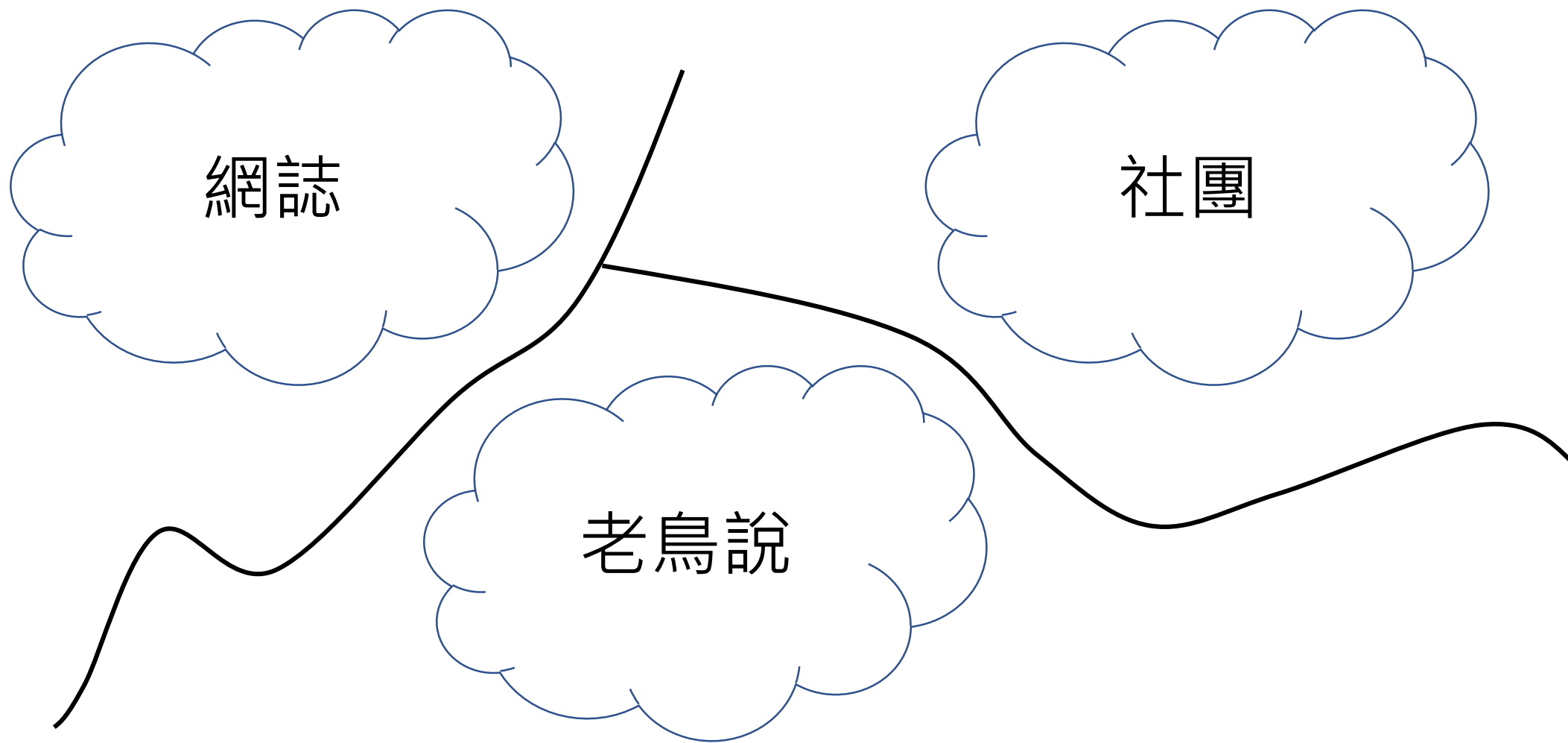


主廚流三層式架構

2.0

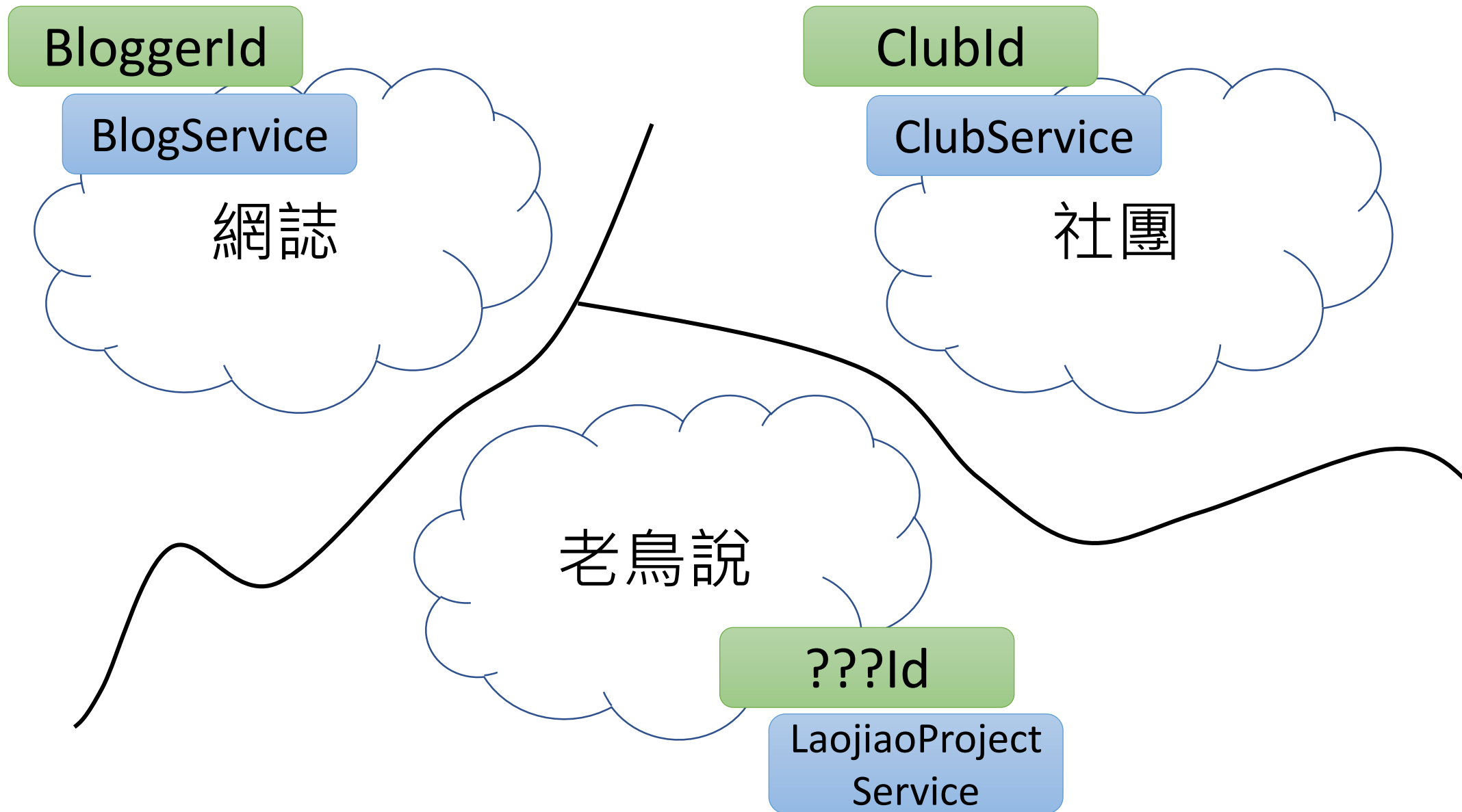
加入邊界的概念

邊界從需求的**語意**中發現，邊界隔起來的地方叫**領域**。



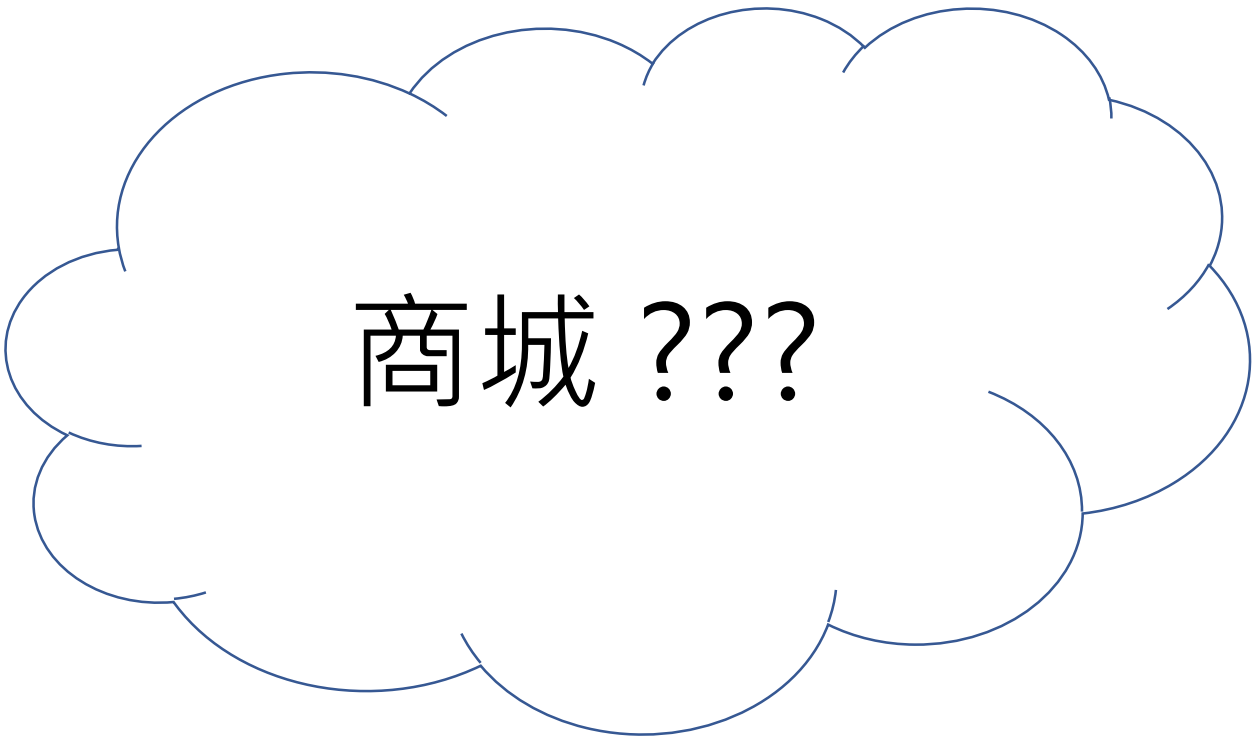
加入邊界的概念

領域必須找到**主要識別**，並建立一個**服務**。



加入邊界的概念

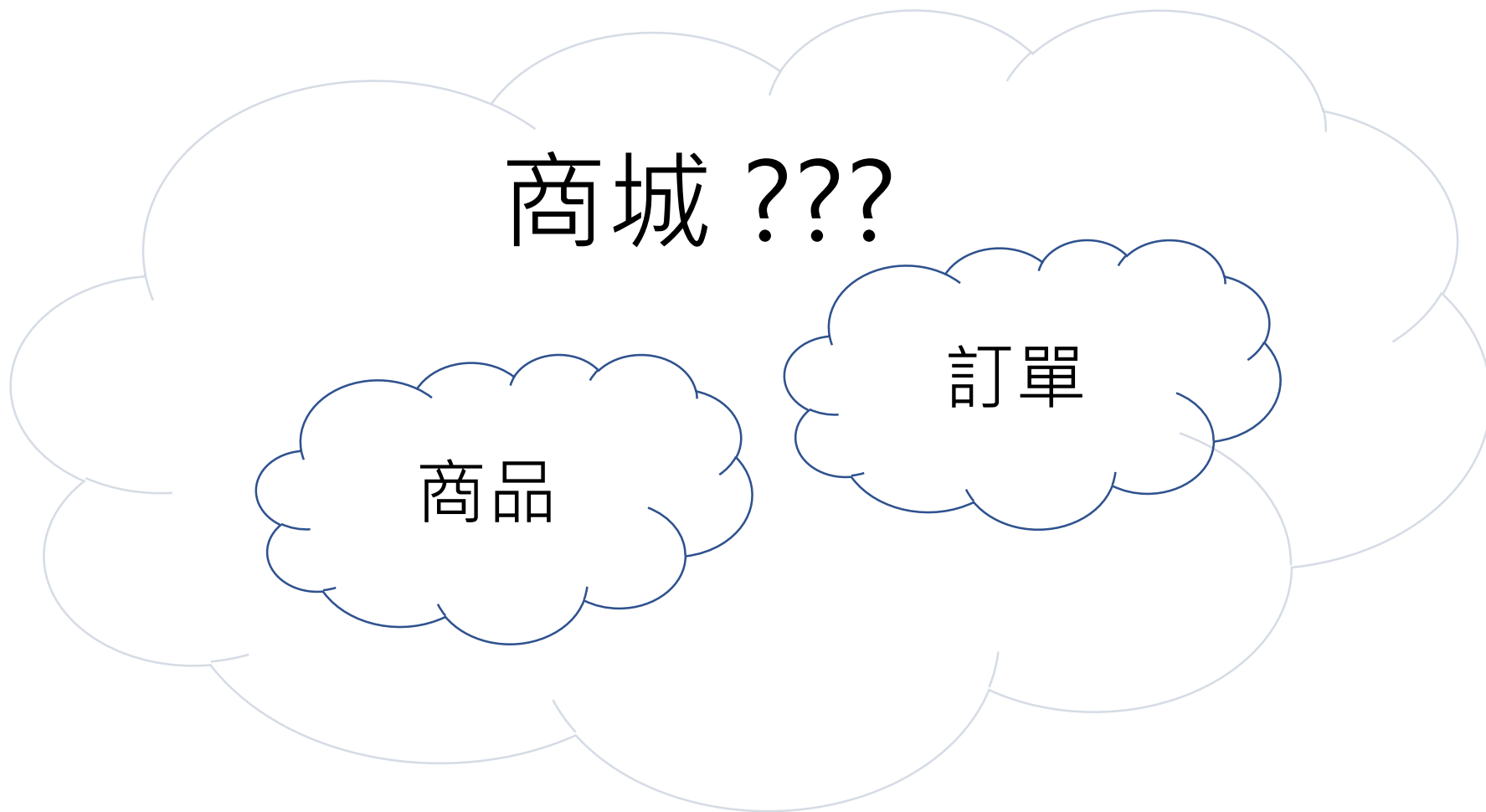
領域必須找到**主要識別**，並建立**一個服務**。



商城 ???

加入邊界的概念

領域必須找到**主要識別**，並建立**一個服務**。



加入邊界的概念

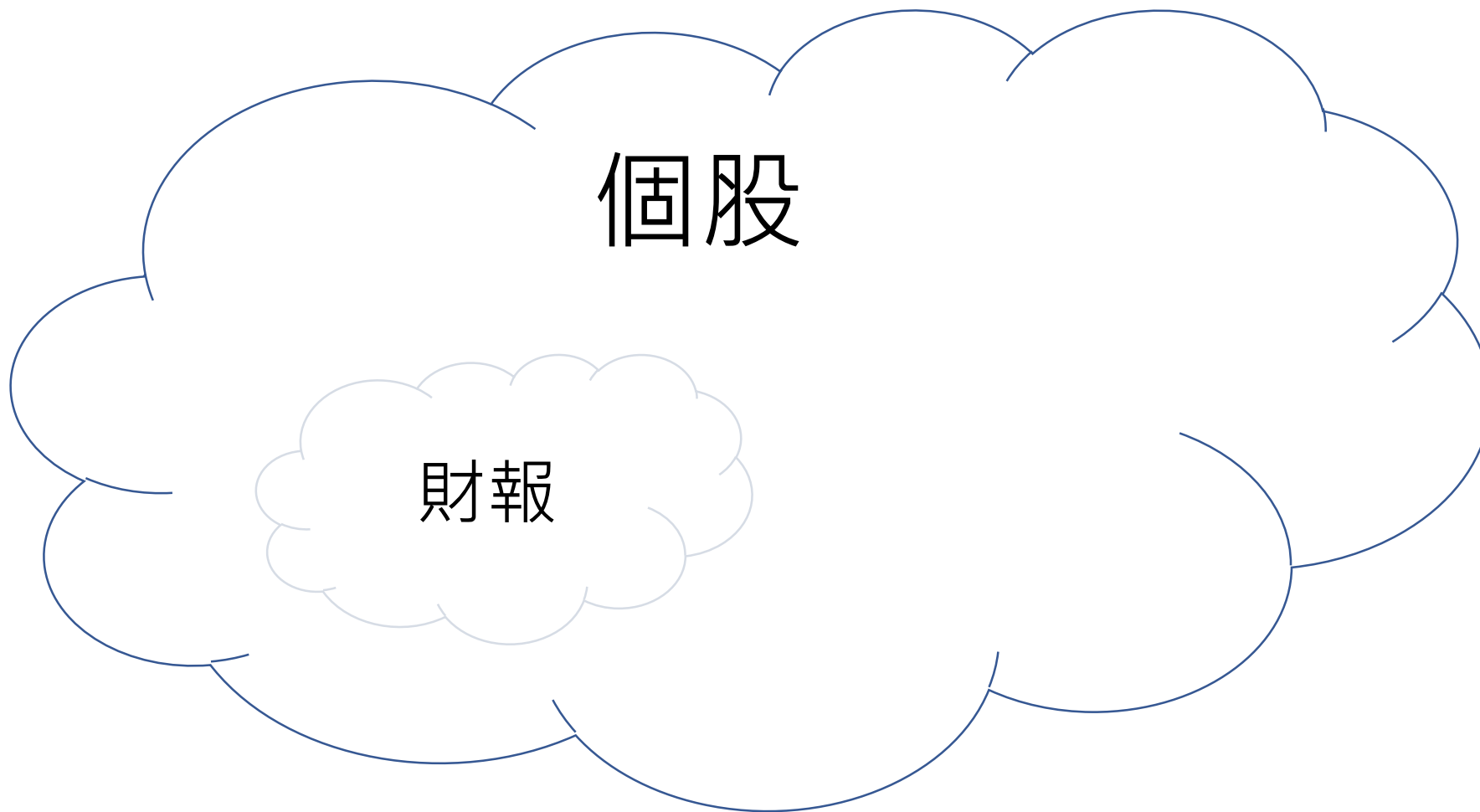
領域必須找到**主要識別**，並建立**一個服務**。



財報 ???

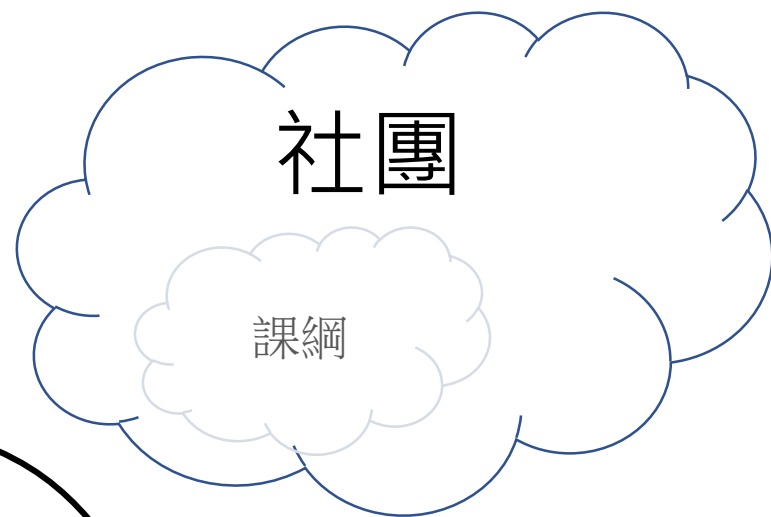
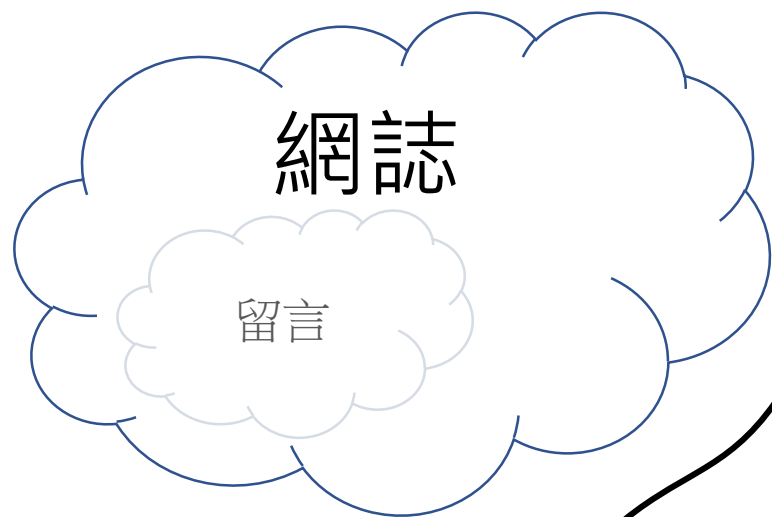
加入邊界的概念

領域必須找到**主要識別**，並建立**一個服務**。



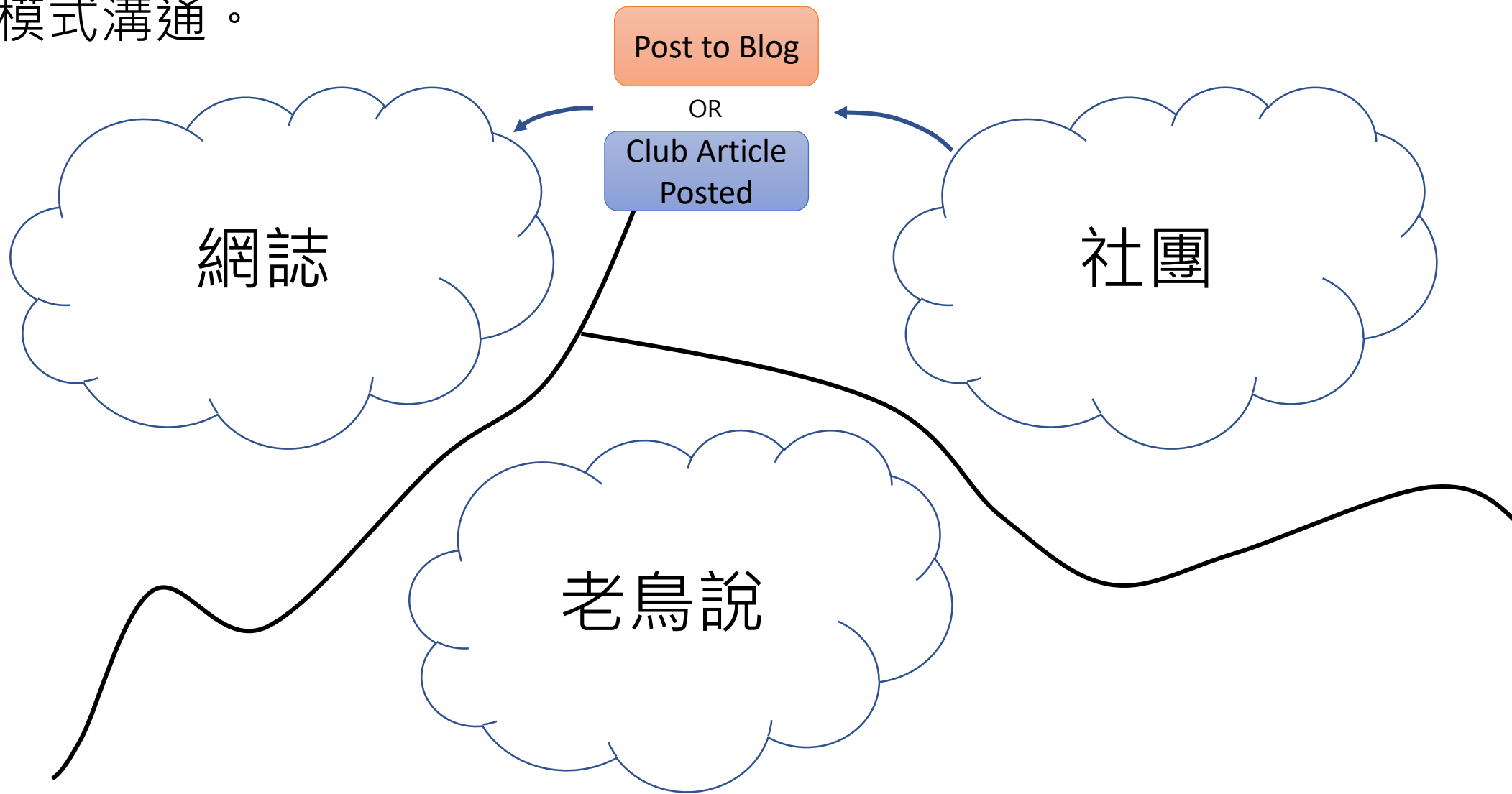
加入邊界的概念

領域還會有子領域

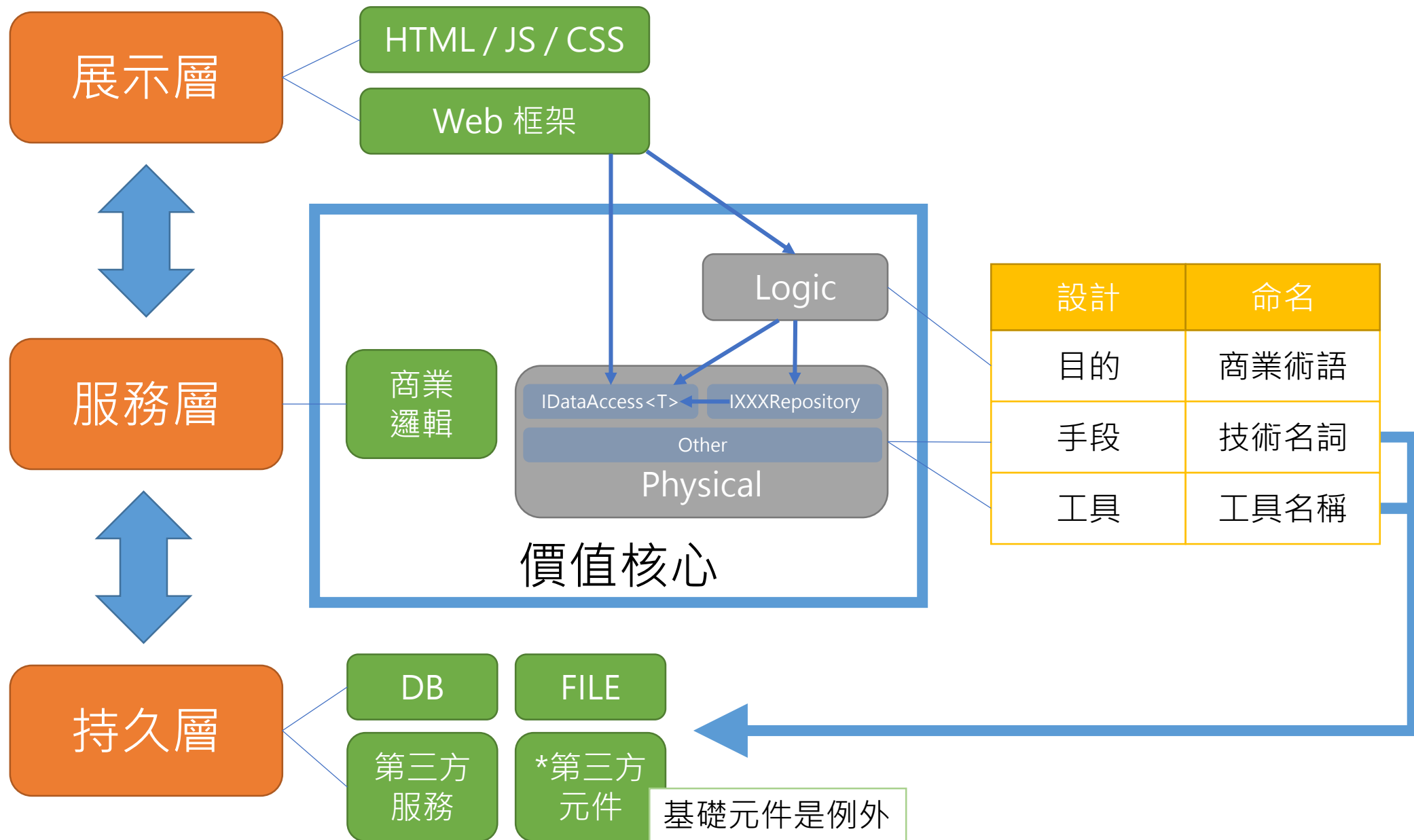


加入邊界的概念

邊界之間不耦合，在更上層耦合，或用 **Command and Event** 模式溝通。



主廚流三層式架構 (Web)




主廚流二層式架構 (Web)

```
public class HomeController : Controller
{
    private readonly ApplicationDbContext _context;

    public HomeController(ApplicationDbContext context)
    {
        this._context = context;
    }

    [HttpGet]
    public IActionResult Index()
    {
        return View();
    }
}
```



主廚流二層式架構 (Web)

```
public class HomeController : Controller
{
    private readonly IDataAccess<Club> clubDataAccess;

    public HomeController(IDataAccess<Club> clubDataAccess)
    {
        this.clubDataAccess = clubDataAccess;
    }

    [HttpGet("{clubId:int}")]
    public async Task<IActionResult> Index(int clubId)
    {
        var club = await this.clubDataAccess.QueryOneAsync(
            x => new { x.Id, x.Name },
            x => x.Id == clubId);

        this.ViewBag.ClubId = club.Id;

        return View();
    }
}
```

主廚流二層式架構 (Web)

更精細地分割**存取資料庫**的職責

IDataAccess<T>

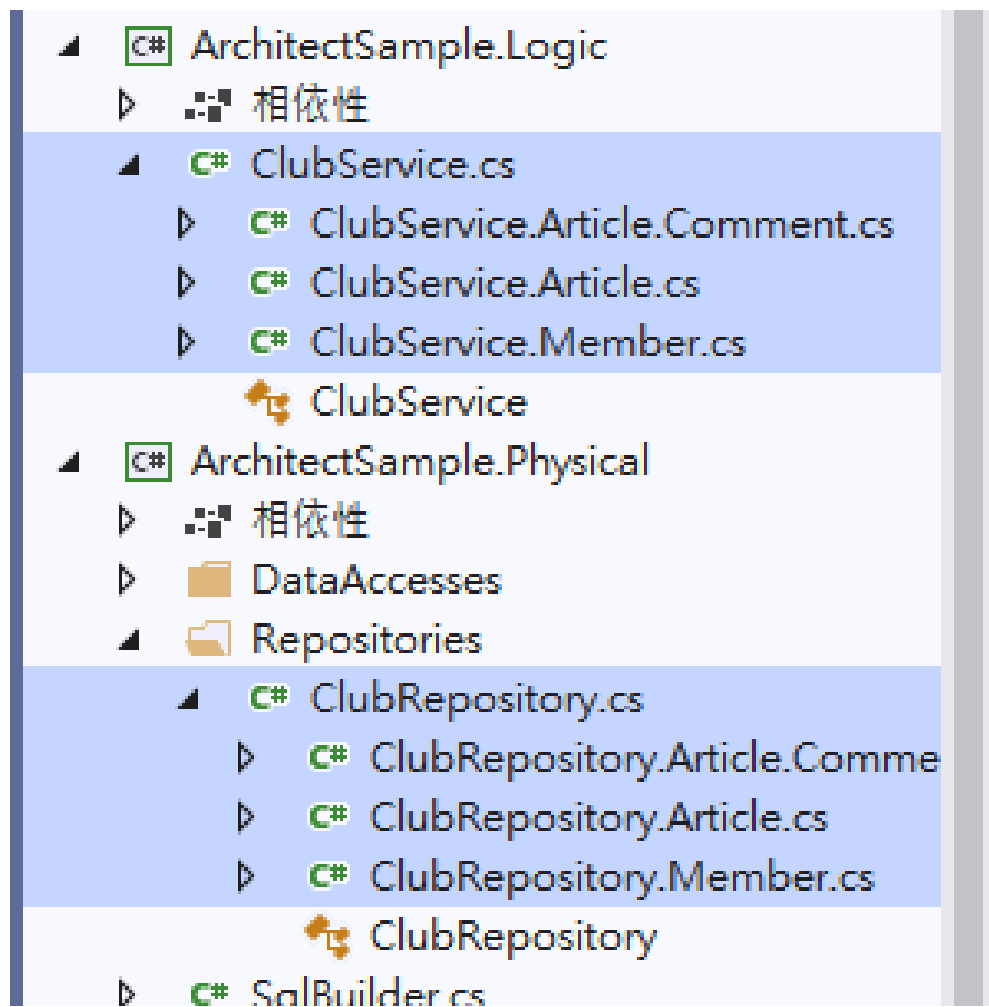
- 一個資料表一個 DataAccess
- 限定實作 IDataAccess<T> 界面
- 使用在對單一資料表簡單的 CRUD

IXXXRepository

- 一個 Service 一個 Repository
- 圍繞「主要識別」的跨資料表操作
- **不處理交易**，交易範圍（ Transaction Scope ）應該要在 Service 定義。

主廚流二層式架構 (Web)

使用 partial class 分割子領域



Let's go DEMO

主廚流2.0五問

1. 當我們發現在 Controller 寫了判斷，卻不是回傳 ActionResult，要想想我們是不是在寫商業邏輯？
2. 當我們發現在 Controller 組合一個以上的 Physical 方法使用，要想想我們是不是在寫商業邏輯？
3. 當我們發現在 Controller 改變了變數的值，要想想我們是不是在寫商業邏輯？
4. 當我們在 Service 只呼叫了 IDataAccess<T> 就回傳，要想想我們是不是沒有商業邏輯？
5. supershowwei/glossary