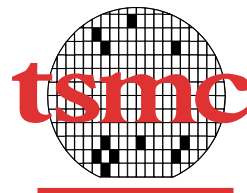




STUDY4

為 學 習 而 生

# 特別感謝



以及各位參與活動的各位



# 系統上線後的問題排查， 反思系統設計

Jimmy Ho

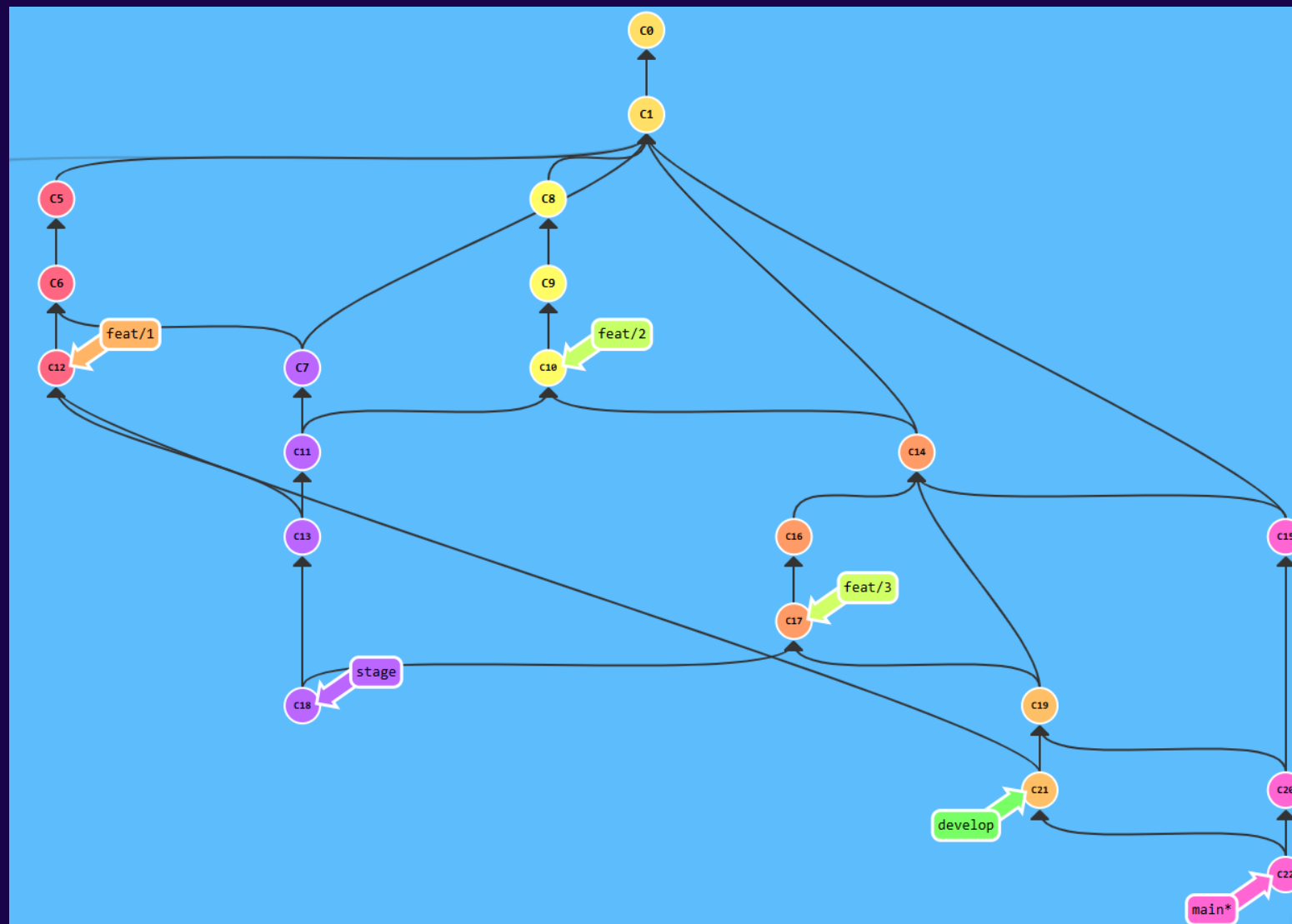


# 今天沒太多程式，著重在情境和思考

隔壁很精彩，還來得及

# 版本管理

# 1. Git 分支管理



# 1. Git 分支管理

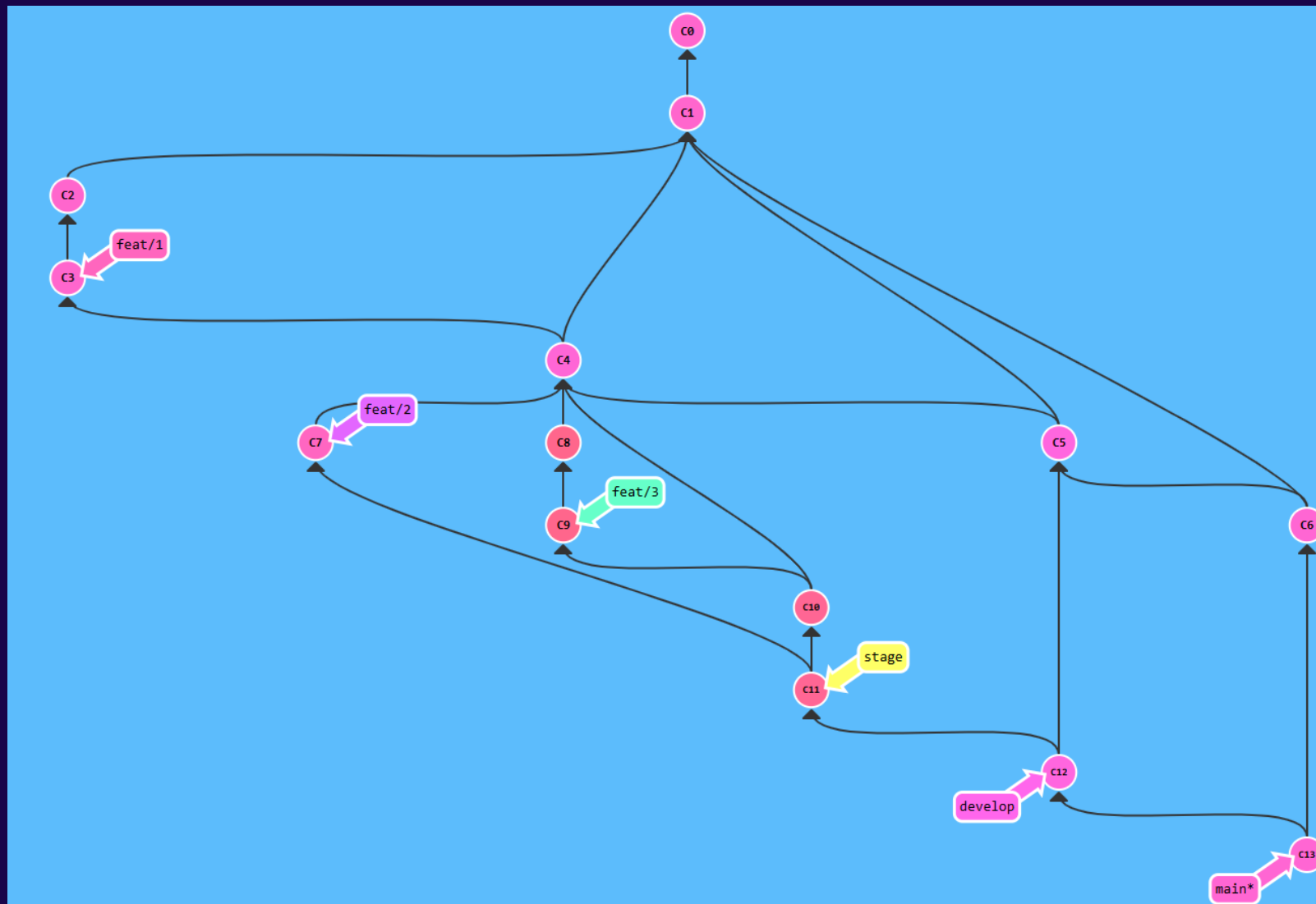
- 容易衝突
- 不好確定 release 的內容是不是都看過
- 容易弄錯分支

# 1. Git 分支管理

- 減少分支
- 隨時都可以合併，減少衝突的產生
- 導入 Feature Toggle 功能
- Toggle 管理問題



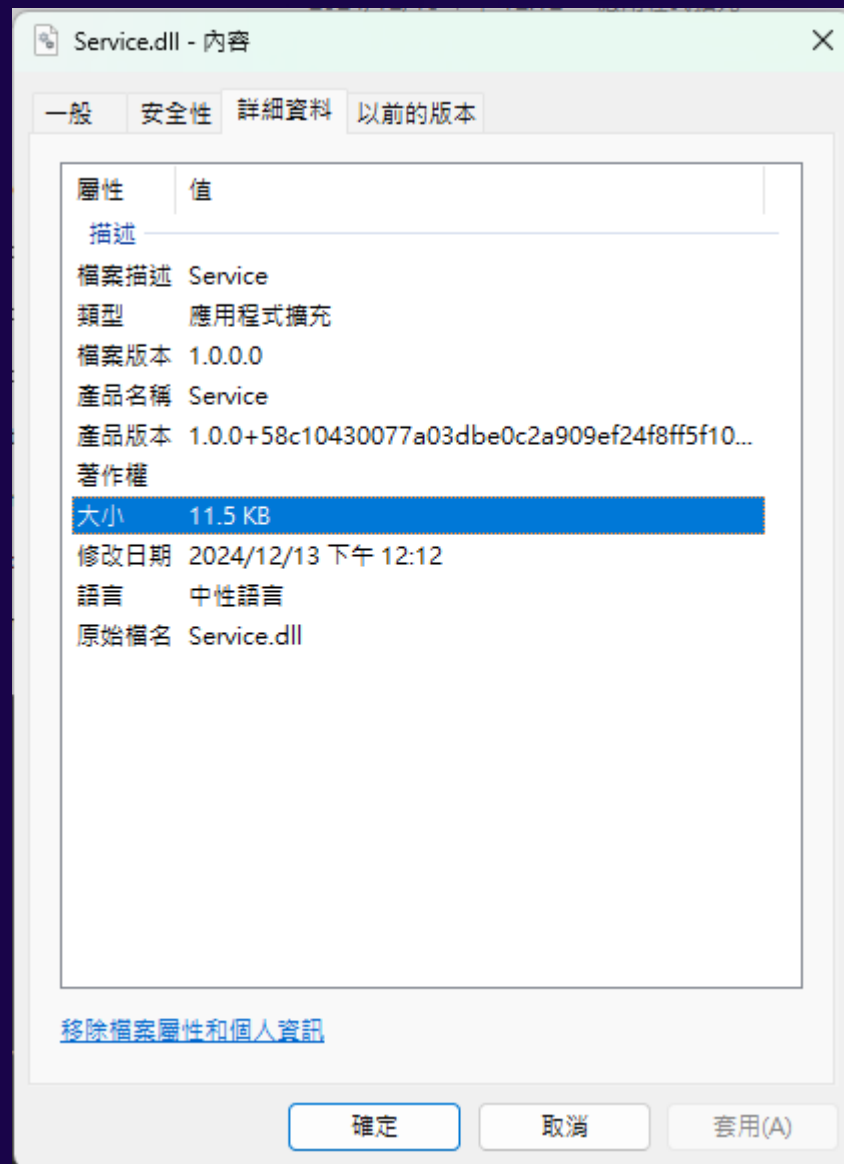
# 1. Git 分支管理



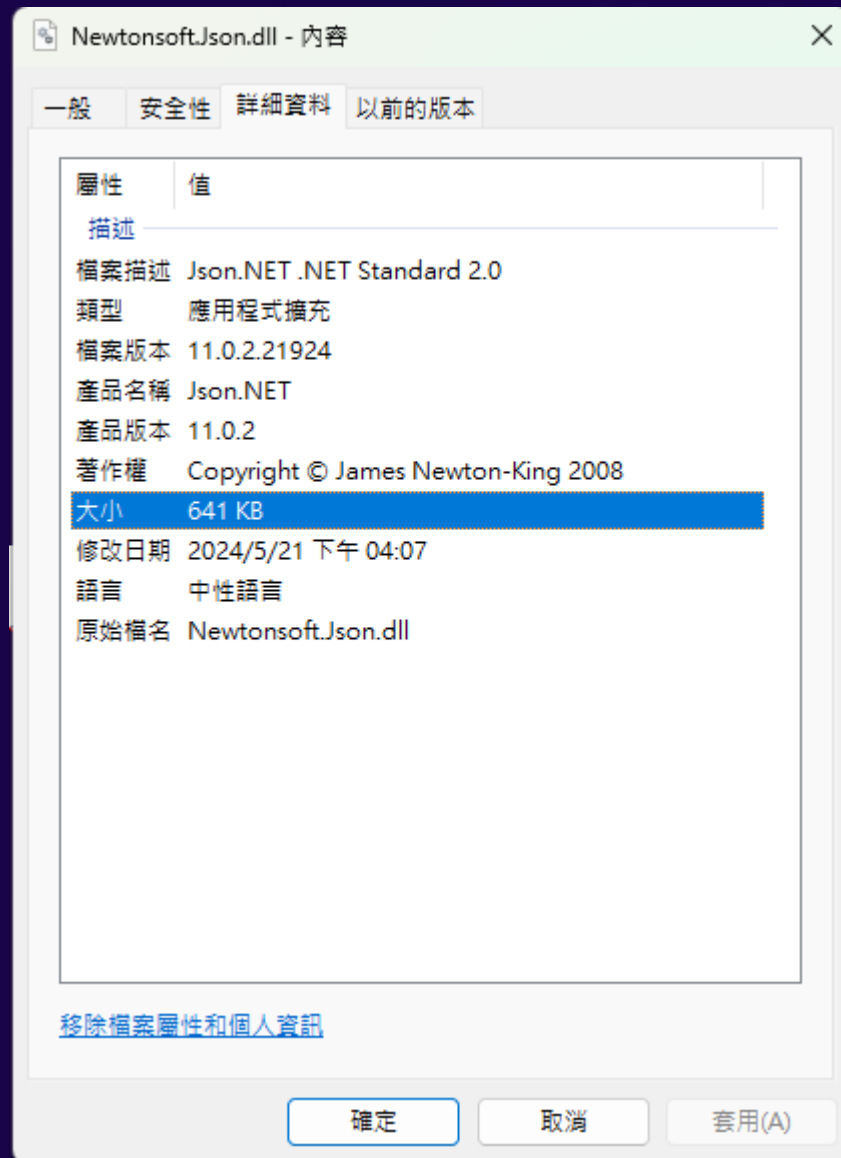
## 2. DII版本管理

- DII放在資料夾做參考
- 客戶反映問題，不知道DII對應的程式版本
- 每個客戶都 Clone 一份程式出來確保不會搞錯版本

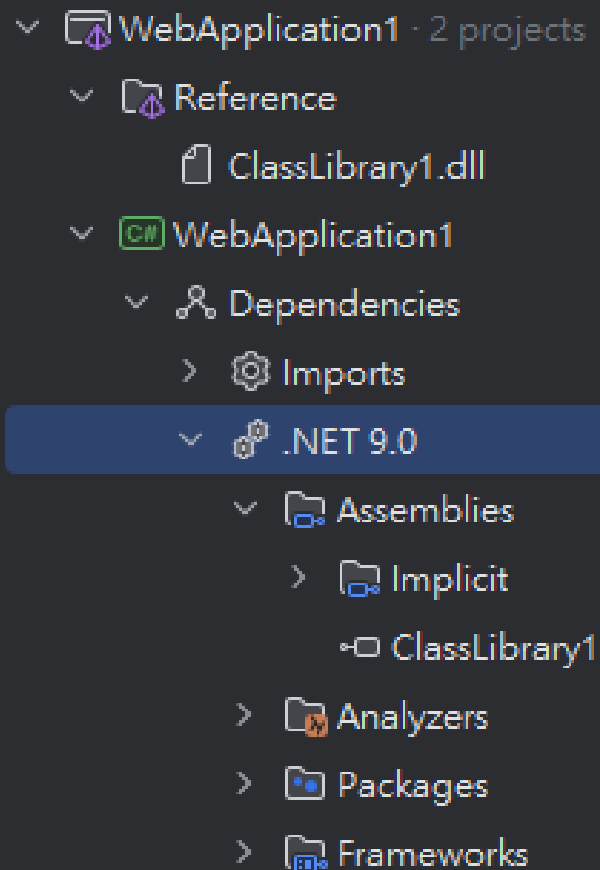
## 2. Dll版本管理



## 2. Dll版本管理



## 2. Dll版本管理



```
<ItemGroup>
  <Reference Include="ClassLibrary1">
    <HintPath>..\Reference\ClassLibrary1.dll</HintPath>
  </Reference>
</ItemGroup>
```

## 2. Dll版本管理

- 每次 Release 更新版號並且上傳到 Nuget
- Dll 參考改為Nuget 參考
- [Blog: 將獨立的 dll 打包發佈到 nuget server](#)

# 資料相關

### 3. 沒有介面

- 遇到問題只能開資料庫直接修改
- 有權限的工程師才能操作
- 很容易漏改資料



### 3. 沒有介面

- 常常修改代表是常態，應該轉換為需求
- 和 PM/主管 協調安排進開發時程
- 避免事情只落在部分人身上

## 4. 資料不足

- 客戶反映資料不是他改的
- 沒有任何紀錄可以查詢

## 4. 資料不足

- 增加歷史紀錄表
- 所有異動都寫入

## 4. 資料不足

資料量比較大，需要思考如何做備份或是切分 (partition)

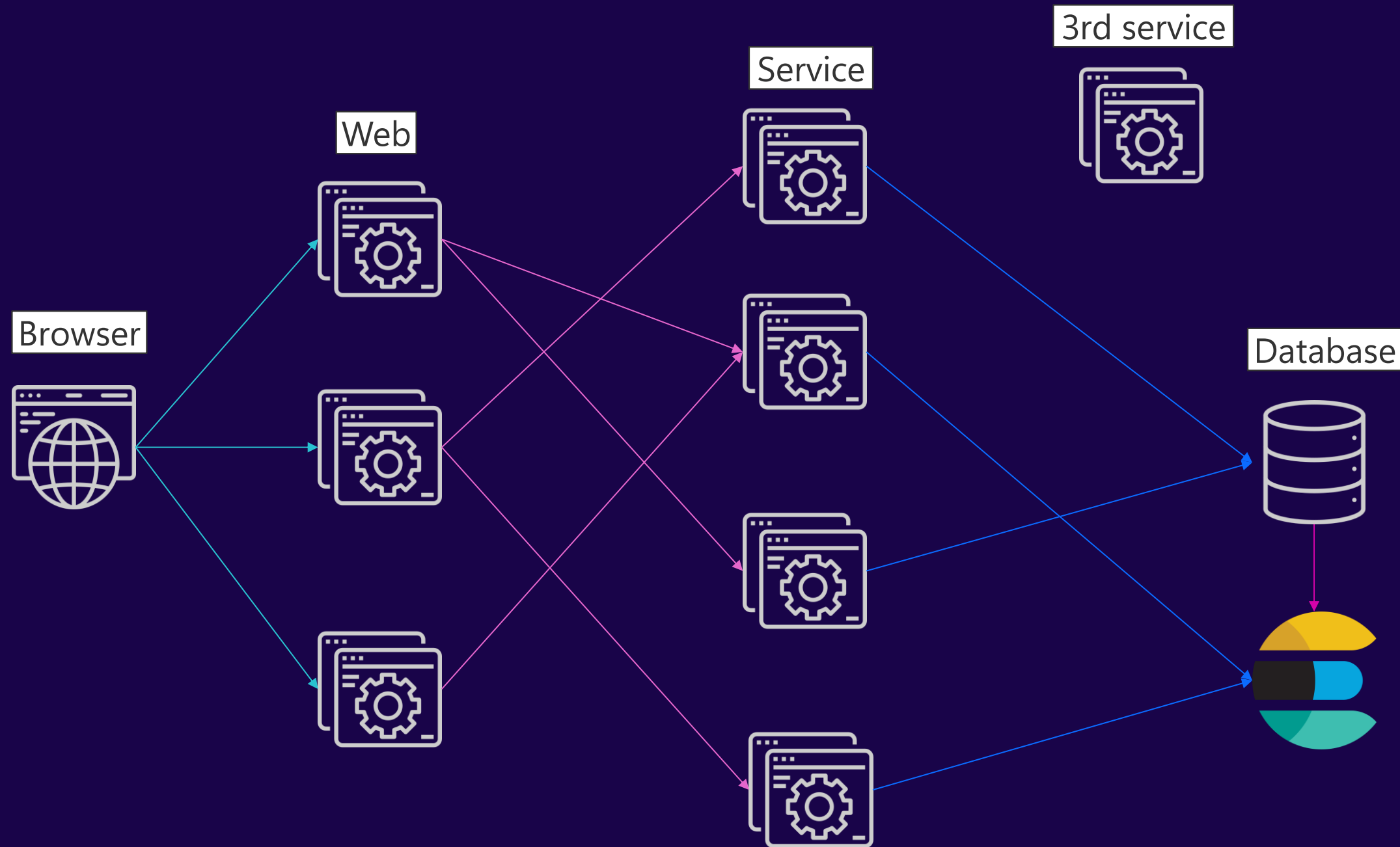
# 設計相關

## 5. Reflection / Design Pattern

- 很好用的功能
- 在排查問題的時候，卻很容易遇到困難
- 很需要 Debug 中斷做確認

## 5. Reflection / Design Pattern

- 重新討論使用的時機
- 在 PR review 要確認是必要的才可以使用





## 6. 分層職責

- Controller/Service/Repository
- 在這樣的架構下，是否需要這樣拆
- Controller/Repository?

## 7. 物件轉換

Controller	Service	Repository
parameter	paramterDto	dataModel

Repository	Service	Controller
dataModel	Dto	viewModel

## 7. 物件轉換

```
public class UserParameter
{
    public string Name { get; set; }
    public int Age { get; set; }
}
```

## 7. 物件轉換

增加一個欄位，要從 service 專案改到 web，  
各 3 層總共要改 12 個類別

## 7. 物件轉換

- 職責分離，避免共用而導致暴露過多資訊
- 類別轉換而產生的維護負擔
- 思考是否需要每個都這樣拆分

## 8. 資料流追蹤

- 客戶回報問題，在沒看程式的情況下，不知道走了哪些路
- 有紀錄 exception 但沒辦法串接上下游

## 8. 資料流追蹤

- 不能只有 exception log 也要 request log
- 加上 Traceld 在 query parameter
- 加上 X-correlation-id 於 header





# Demo

CorrelationId



範例程式



```
public class TraceMiddleware(RequestDelegate next)
{
     Jimmy*
    public async Task InvokeAsync(HttpContext context)
    {
        const string xCorrelationId = "x-correlation-id";
        // 檢查是否已有追蹤 ID，若無則使用 TraceIdentifier
        if (!context.Request.Headers.TryGetValue(xCorrelationId, out var value :StringValues))
        {
            value  = context.TraceIdentifier;
            context.Request.Headers[xCorrelationId] = value;
        }

        context.Response.Headers[xCorrelationId] = value;

        await next(context);
    }
}
```

```
public class CorrelationIdInterceptor(IHttpContextAccessor httpContextAccessor) : DelegatingHandler
{
    👤 Jimmy
    protected override async Task<HttpResponseMessage> SendAsync(HttpRequestMessage request, CancellationToken cancellationToken)
    {
        // 從當前的 HTTP Context 中取得 x-Correlation-id
        const string xCorrelationId = "x-correlation-id";
        var correlationId :string? = httpContextAccessor.HttpContext?.Request.Headers[xCorrelationId].FirstOrDefault();

        // 如果不存在，則使用 TraceIdentifier
        if (string.IsNullOrEmpty(correlationId))
        {
            correlationId = httpContextAccessor.HttpContext?.TraceIdentifier;
        }

        // 加入到 Request Header 中
        request.Headers.Add( name: xCorrelationId, correlationId);

        // 呼叫下一層處理邏輯
        return await base.SendAsync(request, cancellationToken);
    }
}
```

```
builder.Services.AddHttpContextAccessor();
builder.Services.AddTransient<CorrelationIdInterceptor>();
builder.Services.AddHttpClient( name: "test")
    .AddHttpMessageHandler<CorrelationIdInterceptor>();
```



上線後挑戰才是開始

# 我是誰

- Jimmy
- 演講經驗: 2021~2023 Dotnet conf
- 部落格 :  
<https://jiaming0708.github.io/>



# Thank you

