

考題核對截圖

使用框架與語言

- 使用框架: .NET8
- 使用語言: C#

題目背景 (證券場景)

請實作一個「證券交易資料查詢系統」RESTful API，包含股票查詢、下單、委託查詢等

功能。

即時價https://mis.twse.com.tw/stock/api/getStockInfo.jsp?ex_ch=tse_2330.tw

必做

- ✓ • 建立.NET 8 Web API專案 (InMemory儲存) ✓ 2026-02-03

```
<Project Sdk="Microsoft.NET.Sdk.Web">

<PropertyGroup>
  <TargetFramework>net8.0</TargetFramework>
  <Nullable>enable</Nullable>
  <ImplicitUsings>enable</ImplicitUsings>
  <GenerateDocumentationFile>true</GenerateDocumentationFile>
  <NoWarn>$$(NoWarn);1591</NoWarn>
</PropertyGroup>

<ItemGroup>
  <PackageReference Include="FluentValidation" Version="11.10.0" />
  <PackageReference Include="FluentValidation.AspNetCore" Version="11.3.0" />
  <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="8.0.23" />
  <PackageReference Include="Microsoft.EntityFrameworkCore" Version="8.0.11" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="8.0.11" />
  <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
  <PrivateAssets>all</PrivateAssets>
</PackageReference>
<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.11" />
<PackageReference Include="Serilog.AspNetCore" Version="8.0.3" />
<PackageReference Include="Swashbuckle.AspNetCore" Version="6.6.2" />
<PackageReference Include="Swashbuckle.AspNetCore.Annotations" Version="6.5.0" />
</ItemGroup>
```

```
curtiesTradingApi.cs * | o CachedTwseApiClient.cs X
SecuritiesTradingApi > Infrastructure > ExternalApis > CachedTwseApiClient.cs > CachedTwseApiClient
|   using SecuritiesTradingApi.Models.Dtos;
|   using SecuritiesTradingApi.Infrastructure.Cache;

namespace SecuritiesTradingApi.Infrastructure.ExternalApis;

public class CachedTwseApiClient : ITwseApiClient
{
    2 references
    private readonly TwseApiClient _innerClient;
    3 references
    private readonly IMemoryCacheService _cache;
    2 references
    private readonly TimeSpan _cacheDuration;
    1 reference
    private const string CacheKeyPrefix = "StockQuote_";

    0 references
    public CachedTwseApiClient(
        TwseApiClient innerClient,
        IMemoryCacheService cache,
        IConfiguration configuration)
    {
        _innerClient = innerClient;
        _cache = cache;
        _cacheDuration = TimeSpan.FromSeconds(configuration.GetValue<int>("TwseApi:CacheSeconds", 5));
    }
}
```

- ✓ • GET /api/v1/stocks?symbol=&keyword=

```
### 測試 3: 查詢股票 - 使用 keyword 參數搜尋股票簡稱
Send Request
GET {{SecuritiesTradingApi_HostAddress}}/api/v1/stocks?keyword=鴻海
Accept: application/json
o
```

```
1 HTTP/1.1 200 OK
2 Connection: close
3 Content-Type: application/json; charset=utf-8
4 Date: Tue, 03 Feb 2026 06:31:24 GMT
5 Server: Kestrel
6 Transfer-Encoding: chunked
7
8 {
9     "page": 1,
10    "pageSize": 20,
11    "totalCount": 1,
12    "totalPages": 1,
13    "hasPreviousPage": false,
14    "hasNextPage": false,
15    "items": [
16        {
17            "stockCode": "2317",
18            "stockName": "鴻海精密工業股份有限公司",
19            "stockNameShort": "鴻海精密工業股份有限公司",
20            "stockNameEn": "鴻海",
21            "exchange": "TWSE",
22            "industry": "31",
23            "lotSize": 1000,
24            "allowOddLot": false,
25            "isActive": true,
26            "listedDate": "2026-02-01T00:00:00"
27        }
28    ]
}
```

- GET /api/v1/stocks/{symbol}

```
### 測試 4: 查詢特定股票基本資料 (台積電)
```

Send Request

```
GET {{SecuritiesTradingApi_HostAddress}}/api/v1/stocks/2330
```

```
Accept: application/json
```

```
1 HTTP/1.1 200 OK
2 Connection: close
3 Content-Type: application/json; charset=utf-8
4 Date: Tue, 03 Feb 2026 06:30:08 GMT
5 Server: Kestrel
6 Transfer-Encoding: chunked

7
8 <{
9   "stockCode": "2330",
10  "stockName": "台灣積體電路製造股份有限公司",
11  "stockNameShort": "台灣積體電路製造股份有限公司",
12  "stockNameEn": "台積電",
13  "exchange": "TWSE",
14  "industry": "24",
15  "lotSize": 1000,
16  "allowOddLot": false,
17  "isActive": true,
18  "listedDate": "2026-02-01T00:00:00"
19 }
```

- POST /api/v1/orders 建立委託單 (買/賣、價格、數量)

```
### 測試 6: 建立限價買單 (台積電)
```

Send Request

```
POST {{SecuritiesTradingApi_HostAddress}}/api/v1/orders
```

```
Content-Type: application/json
```

```
{
  "userId": 1,
  "stockCode": "2330",
  "orderType": 1,
  "buySell": 1,
  "price": 1785.0,
  "quantity": 1000
}
```

```
1 HTTP/1.1 201 Created
2 Connection: close
3 Content-Type: application/json; charset=utf-8
4 Date: Tue, 03 Feb 2026 06:32:23 GMT
5 Server: Kestrel
6 Location: http://localhost:5205/api/v1/Orders?id=1000000016
7 Transfer-Encoding: chunked
8
9 <{
10  "orderId": 1000000016,
11  "stockCode": "2330",
12  "stockName": "台灣積體電路製造股份有限公司",
13  "orderType": 1,
14  "orderTypeName": "Limit",
15  "buySell": 1,
16  "buySellName": "Buy",
17  "price": 1785.0,
18  "quantity": 1000,
19  "orderStatus": 1,
20  "orderStatusName": "Pending",
21  "tradeDate": "2026-02-03T00:00:00Z",
22  "createdAt": "2026-02-03T06:32:23.7602717Z"
23 }
```

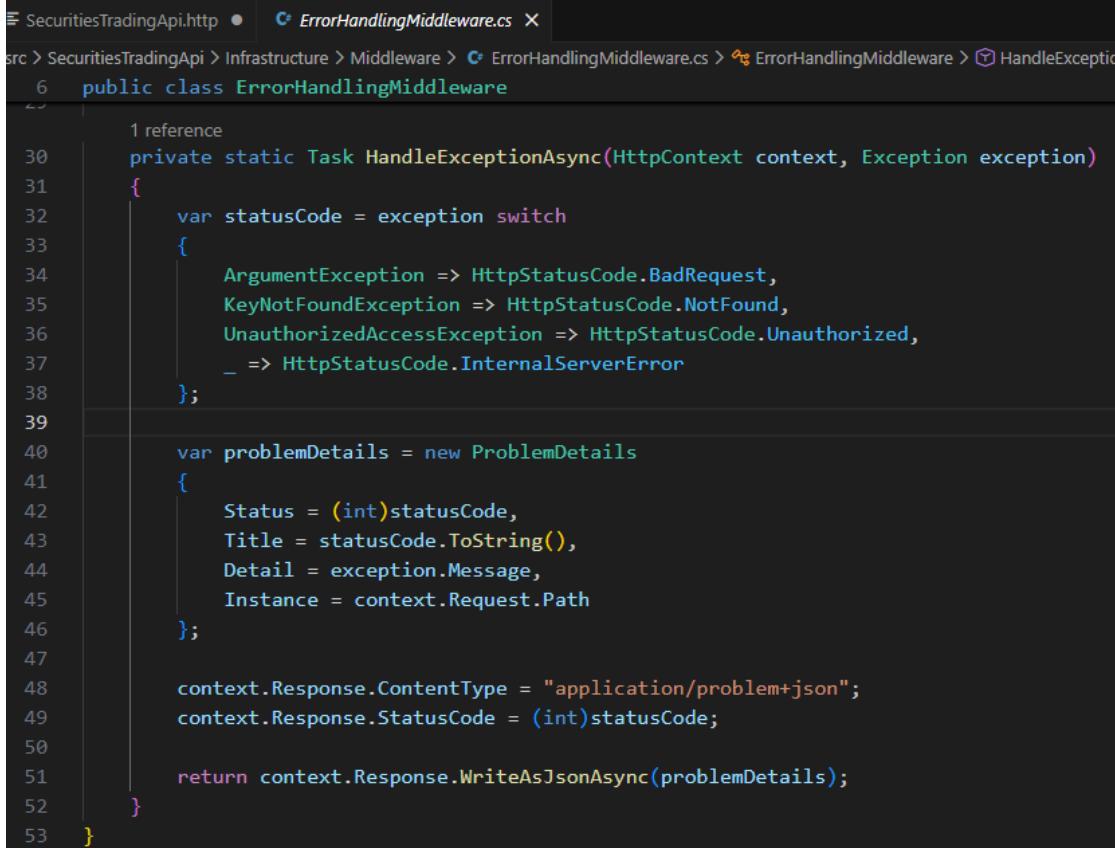
- GET /api/v1/orders/{id}

測試 7：查詢特定委託單詳細資料

Send Request
GET {{SecuritiesTradingApi_HostAddress}}/api/v1/orders/1000000010
Accept: application/json

```
1 HTTP/1.1 200 OK
2 Connection: close
3 Content-Type: application/json; charset=utf-8
4 Date: Tue, 03 Feb 2026 06:33:21 GMT
5 Server: Kestrel
6 Transfer-Encoding: chunked
7
8 {
9     "orderId": 1000000010,
10    "userId": 1,
11    "userName": null,
12    "stockCode": "2330",
13    "stockName": "台積電",
14    "stockNameShort": "台積電",
15    "orderType": 1,
16    "orderTypeName": "Buy",
17    "buySell": 1,
18    "buySellName": null,
19    "price": 1785.00,
20    "quantity": 1000,
21    "filledQuantity": 0,
22    "orderStatus": 1,
23    "orderStatusName": "Pending",
24    "tradeDate": "2026-02-03T00:00:00",
25    "createdAt": "2026-02-03T03:56:25.9099574"
26 }
```

- Model驗證與錯誤處理 (400/404)



The screenshot shows a code editor with the file `ErrorHandlingMiddleware.cs` open. The code defines a middleware class that handles exceptions by returning appropriate HTTP status codes and problem details.

```
6 public class ErrorHandlingMiddleware
  ~~~
  1 reference
30     private static Task HandleExceptionAsync(HttpContext context, Exception exception)
31     {
32         var statusCode = exception switch
33         {
34             ArgumentException => HttpStatusCode.BadRequest,
35             KeyNotFoundException => HttpStatusCode.NotFound,
36             UnauthorizedAccessException => HttpStatusCode.Unauthorized,
37             _ => HttpStatusCode.InternalServerError
38         };
39
40         var problemDetails = new ProblemDetails
41         {
42             Status = (int)statusCode,
43             Title = statusCode.ToString(),
44             Detail = exception.Message,
45             Instance = context.Request.Path
46         };
47
48         context.Response.ContentType = "application/problem+json";
49         context.Response.StatusCode = (int)statusCode;
50
51         return context.Response.WriteAsJsonAsync(problemDetails);
52     }
53 }
```

- 啟用Swagger

Securities Trading API v1 OAS 3.0

<http://localhost:5205/swagger/v1/swagger.json>

證券交易資料查詢系統 API - 提供台灣股票資訊查詢和委託下單功能

[Contact Securities Trading API Team](#)

Orders

<code>GET</code>	<code>/api/v1/Orders</code>	查詢委託單列表	▼
<code>POST</code>	<code>/api/v1/Orders</code>	建立股票委託單	▼
<code>GET</code>	<code>/api/v1/Orders/{orderId}</code>	查詢委託單詳細資料	▼

Stocks

<code>GET</code>	<code>/api/v1/Stocks</code>	查詢股票列表 (支援symbol或keyword查詢)	▼
<code>GET</code>	<code>/api/v1/Stocks/{symbol}</code>	查詢股票基本資料	▼
<code>GET</code>	<code>/api/v1/Stocks/{symbol}/Info</code>	查詢股票即時報價	▼

- ✓ 分層架構 輕量級CQRS：資料庫部分先暫時進行以利未來拆分更完整的分層，如未來 寫跟讀的比例差異太大在進行切割

C#

```
// OrderService.cs
// Write to Orders_Write (hot layer) - 寫入優化表
var orderWrite = new OrdersWrite
{
    OrderId = orderId,
    UserId = orderDto.UserId,
    StockCode = orderDto.StockCode,      // ⚠ 只存 StockCode (正規化)
    OrderType = orderDto.OrderType,      // ⚠ 只存代碼, 不存名稱
    BuySell = orderDto.BuySell,
    Price = orderDto.Price,
    Quantity = orderDto.Quantity,
    OrderStatus = 1,
    TradeDate = tradeDate,
    CreatedAt = createdAt,
    OrderSeq = orderId
};
_context.OrdersWrite.Add(orderWrite); // 👍 寫入命令模型

// Sync to Orders_Read (warm layer) - 讀取優化表
var orderRead = new OrdersRead
{
    OrderId = orderId,
    // ... 基本欄位 ...
    StockName = stock.StockName,          // ✅ 反正規化 - 避免 JOIN
    StockNameShort = stock.StockNameShort, // ✅ 反正規化
    OrderTypeName = orderDto.OrderType == 1 ? "Limit" : "Market", // ✅ 反正規化
    BuySellName = orderDto.BuySell == 1 ? "Buy" : "Sell",           // ✅ 反正規化
    OrderStatusName = "Pending",          // ✅ 反正規化
    // ...
};
_context.OrdersRead.Add(orderRead); // 👍 同步到查詢模型

// ✅ 合併為一次保存 (改善性能)
await _context.SaveChangesAsync(cancellationToken); // 👍 單一交易, 保證一致性

//TradingDbContext.cs
public DbSet<OrdersWrite> OrdersWrite { get; set; } = null!; // 👍 寫入表
public DbSet<OrdersRead> OrdersRead { get; set; } = null!; // 👍 讀取表
```

- ✓ 分頁查詢(股票列表) ✓ 2026-02-03

Page 1

```
### 測試 3-1: 查詢股票 - 分頁查詢（預設第1頁，每頁20筆）
Send Request
GET {{SecuritiesTradingApi_HostAddress}}/api/v1/stocks?keyword=電
Accept: application/json
```

```
1 HTTP/1.1 200 OK
2 Connection: close
3 Content-Type: application/json; charset=utf-8
4 Date: Tue, 03 Feb 2026 06:38:54 GMT
5 Server: Kestrel
6 Transfer-Encoding: chunked
7
8 {
9   "page": 1,
10  "pageSize": 20,
11  "totalCount": 152,
12  "totalPages": 8,
13  "hasPreviousPage": false,
14  "hasNextPage": true,
15  "items": [
16    {
17      "stockCode": "1503",
18      "stockName": "士林電機廠股份有限公司",
19      "stockNameShort": "士林電機廠股份有限公司",
20      "stockNameEn": "士電",
21      "exchange": "TWSE",
22      "industry": "05",
23      "lotSize": 1000,
24      "allowOddLot": false,
25      "isActive": true,
26      "listedDate": "2026-02-01T00:00:00"
27    },
28  ],
29  "links": [
30    {
31      "url": "/api/v1/stocks?page=2&keyword=%E9%9B%86",
32      "label": "Next"
33    }
34  ]
35}
```

Page 2

```
### 測試 3-2: 查詢股票 - 分頁查詢（第2頁，預設每頁20筆）
Send Request
GET {{SecuritiesTradingApi_HostAddress}}/api/v1/stocks?keyword=電&page=2
Accept: application/json
```

```
1 HTTP/1.1 200 OK
2 Connection: close
3 Content-Type: application/json; charset=utf-8
4 Date: Tue, 03 Feb 2026 06:39:24 GMT
5 Server: Kestrel
6 Transfer-Encoding: chunked
7
8 {
9   "page": 2,
10  "pageSize": 20,
11  "totalCount": 152,
12  "totalPages": 8,
13  "hasPreviousPage": true,
14  "hasNextPage": true,
15  "items": [
16    {
17      "stockCode": "2248",
18      "stockName": "華勝汽車電子股份有限公司",
19      "stockNameShort": "華勝汽車電子股份有限公司",
20      "stockNameEn": "華勝-KY",
21      "exchange": "TWSE",
22      "industry": "12",
23      "lotSize": 1000,
24      "allowOddLot": false,
25      "isActive": true,
26      "listedDate": "2026-02-01T00:00:00"
27    },
28  ],
29  "links": [
30    {
31      "url": "/api/v1/stocks?page=1&keyword=%E9%9B%86",
32      "label": "Previous"
33    },
34    {
35      "url": "/api/v1/stocks?page=3&keyword=%E9%9B%86",
36      "label": "Next"
37    }
38  ]
39}
```

• 單元測試 2026-02-03

The screenshot shows the Visual Studio Test Results window for a unit test run on 2026-02-03. The results are as follows:

- PROBLEMS: 180
- OUTPUT
- DEBUG CONSOLE
- TERMINAL
- TEST RESULTS** (selected)
- PORTS
- NUGET

Running tests...

PASSED: Test1 (1ms)
PASSED: GetStockInfo_ExistingStock_ReturnsOk (1401ms)
PASSED: GetStockInfo_NonExistingStock_ReturnsNotFound (67ms)
PASSED: CreateOrder_ValidOrder_ReturnsCreated (1596ms)
PASSED: CreateOrder_InvalidQuantity_ReturnsBadRequest (10ms)
PASSED: GetOrder_NonExistingOrder_ReturnsNotFound (16ms)
PASSED: CreateOrder_NonExistingStock_ReturnsNotFound (4ms)
PASSED: GetStockQuote_ExistingStock_ReturnsOkOrServiceUnavailable (185ms)

=====

Test Run Summary

=====

Total Tests: 8
Passed: 8
Failed: 0
Skipped: 0
Result: SUCCEEDED
Test Execution Time: 1.78s (1783ms)
Total Time (including build): 4.75s (4752ms)

=====

C# Dev Kit

- CreateOrder_ValidOrder_ReturnsCreated OrdersControllerTest...
- GetStockInfo_ExistingStock_ReturnsOk StocksControllerTests < ...
- Test1 UnitTest1 + SecuritiesTradingApi.IntegrationTests + SecuritiesT...
- GetStockInfo_NonExistingStock_ReturnsNotFound StocksCon...
- CreateOrder_InvalidQuantity_ReturnsBadRequest OrdersCont...
- GetOrder_NonExistingOrder_ReturnsNotFound OrdersControl...
- CreateOrder_NonExistingStock_ReturnsNotFound OrdersCon...
- GetStockQuote_ExistingStock_ReturnsOkOrServiceUnavailabl...

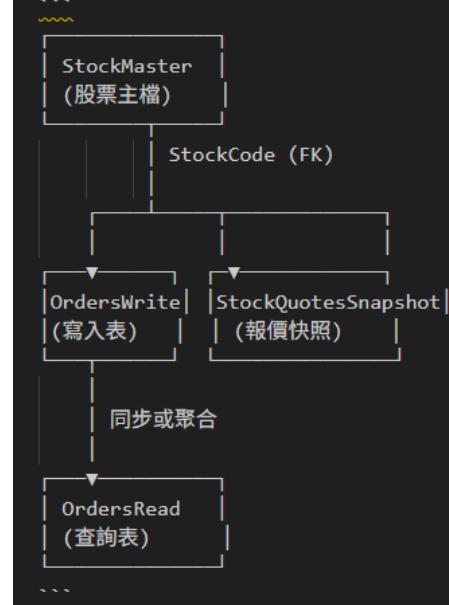
> 2 older results

• 資料庫結構設計

資料庫架構概覽



資料流與關聯



- Log 使用與設計 ✅ 2026-02-03

```

"Serilog": [
  "Using": ["Serilog.Sinks.Console", "Serilog.Sinks.File"],
  "MinimumLevel": "Information",
  "Default": "Information",
  "Override": {
    "Microsoft": "Warning",
    "Microsoft.AspNetCore": "Warning",
    "System": "Warning"
  },
  "WriteTo": [
    {
      "Name": "Console",
      "Args": {
        "outputTemplate": "[{Timestamp:HH:mm:ss} {Level:u3}] {Message:lj} {Properties:j}{NewLine}{Exception}"
      }
    },
    {
      "Name": "File",
      "Args": {
        "path": "logs/app.log",
        "rollingInterval": "Day",
        "retainedFileCountLimit": 30,
        "outputTemplate": "[{Timestamp:yyyy-MM-dd HH:mm:ss.fff zzz} {Level:u3}] {Message:lj} {Properties:j}{NewLine}{Exception}"
      }
    }
  ]
}
  
```

```

securitiesTradingApi[http://.../app-20260203.log] • F app-20260203.log
[2026-02-03 08:57:13.841 +00:00] [INF] Now listening on: http://localhost:5005 ([EventId:{ "id":15, "name": "ListeningOnAddress" }], [SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:57:13.841 +00:00] [INF] Application started. Press Ctrl+C to shut down. ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:57:13.864 +00:00] [INF] Hosting environment: Development ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:57:13.866 +00:00] [INF] Content root path: D:\WebStock\230\src\SecurityTradingApi ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:57:32.403 +00:00] [INF] Application is shutting down... ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:57:41.391 +00:00] [INF] Application is shutting down... ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:57:41.391 +00:00] [INF] Application started. Press Ctrl+C to shut down. ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:57:41.393 +00:00] [INF] Hosting environment: Development ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:57:41.439 +00:00] [INF] Application is shutting down... ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:57:41.441 +00:00] [INF] Content root path: D:\WebStock\230\src\SecurityTradingApi ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:58:04.456 +00:00] [INF] Application is shutting down... ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:58:14.903 +00:00] [INF] Application is shutting down... ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:58:14.967 +00:00] [INF] Application started. Press Ctrl+C to shut down. ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
[2026-02-03 08:58:14.969 +00:00] [INF] Hosting environment: Development ([SourceContext:"Microsoft.Hosting.Lifetime", "Application": "SecurityTradingApi"])
  
```

- Middleware or filter 實作統一回傳格式 使用Middleware針對錯誤格式進行統一回傳格式

```

stress-test.js M ✘ ErrorHandlingMiddleware.cs ✘ stress-test-report.md 9+, M JS load-test.js
  
```

SecurityTradingApi > Infrastructure > Middleware > ErrorHandlingMiddleware.cs

```

public class ErrorHandlingMiddleware
{
    private static Task HandleExceptionAsync(HttpContext context, Exception exception)
    {
        var statusCode = exception switch
        {
            ArgumentException => HttpStatusCode.BadRequest,
            KeyNotFoundException => HttpStatusCode.NotFound,
            UnauthorizedAccessException => HttpStatusCode.Unauthorized,
            _ => HttpStatusCode.InternalServerError
        };

        var problemDetails = new ProblemDetails
        {
            Status = (int)statusCode,
            Title = statusCode.ToString(),
            Detail = exception.Message,
            Instance = context.Request.Path
        };

        context.Response.ContentType = "application/problem+json";
        context.Response.StatusCode = (int)statusCode;
    }

    return context.Response.WriteAsJsonAsync(problemDetails);
}
  
```

- 快取 ✅ 2026-02-03

```
MemoryCacheService.cs  CachedTwseApiClient.cs X

src\SecuritiesTradingApi\Infrastructure\ExternalApis> CachedTwseApiClient.cs > CachedTwseApiClient

  1 using SecuritiesTradingApi.Models.Dtos;
  2 using SecuritiesTradingApi.Infrastructure.Cache;
  3
  4 namespace SecuritiesTradingApi.Infrastructure.ExternalApis;
  5
  6     2 references
  7     public class CachedTwseApiClient : ITwseApiClient
  8     {
  9         2 references
 10         private readonly TwseApiClient _innerClient;
 11         3 references
 12         private readonly IMemoryCacheService _cache;
 13         2 references
 14         private readonly TimeSpan _cacheDuration;
 15         1 reference
 16         private const string CacheKeyPrefix = "StockQuote_";
 17
 18         0 references
 19         public CachedTwseApiClient(
 20             TwseApiClient innerClient,
 21             IMemoryCacheService cache,
 22             IConfiguration configuration)
 23         {
 24             _innerClient = innerClient;
 25             _cache = cache;
 26             _cacheDuration = TimeSpan.FromSeconds(configuration.GetValue<int>("TwseApi:CacheSeconds", 5));
 27         }
 28     }
 29 }
```

```
    /> references
    public async Task<StockInfoDto> GetStockInfoAsync(string stockCode, CancellationToken cancellationToken = default)
    {
        _logger.LogInformation("Querying stock info for {StockCode}", stockCode);

        // Try to get from memory cache first
        var cacheKey = $"stock:info:{stockCode}";
        var cachedInfo = _cacheService.Get<StockInfoDto>(cacheKey);

        if (cachedInfo != null)
        {
            _logger.LogInformation("Stock info for {StockCode} retrieved from cache", stockCode);
            return cachedInfo;
        }

        _logger.LogInformation("Cache miss for stock info {StockCode}, fetching from database", stockCode);

        var stock = await _context.StockMaster
            .AsNoTracking()
            .FirstOrDefaultAsync(s => s.StockCode == stockCode, cancellationToken);

        if (stock == null)
        {
            _logger.LogWarning("Stock {StockCode} not found", stockCode);
            return null;
        }
    }
}
```

- 驗證實作

GET /api/v1/Orders/{orderId} 查詢委託單詳細資料

Parameters

Name	Description
orderId * required integer(\$int64) (path)	委託單編號 1000000577

Execute

Responses

Curl

```
curl -X 'GET' \
'http://localhost:5205/api/v1/Orders/1000000577' \
-H 'accept: */*' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOj8v...
```

Request URL

```
http://localhost:5205/api/v1/Orders/1000000577
```

Server response

Curl

```
curl -X 'GET' \
'http://localhost:5205/api/v1/Orders/1000000577' \
-H 'accept: */*' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOj8v...
```

Request URL

```
http://localhost:5205/api/v1/Orders/1000000577
```

Server response

Code **Details**

200 Response body

```
{
  "orderId": 1000000577,
  "userId": 2,
  "userName": null,
  "stockCode": "2454",
  "stockName": null,
  "stockNameShort": null,
  "orderType": 1,
  "orderTypeName": null,
  "buySell": 2,
  "buySellName": null,
  "price": 93.5,
  "quantity": 1000,
  "filledQuantity": 0,
  "orderStatus": 1,
  "orderStatusName": null,
  "tradeDate": "2026-02-03T18:44:04.7966667",
  "createdAt": "2026-01-30T10:44:04.7966667"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Tue, 03 Feb 2026 10:59:48 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code **Description**

200 成功返回委託單資料

✓ • 壓力測試腳本撰寫 ✓ 2026-02-03

壓力測試(Stress)



```
execution: local
script: ./stress-test.js
output: -
```

```
scenarios: {100.00%} 1 scenario, 300 max VUs, 1m0s max duration (incl. graceful stop):
  * default: Up to 300 looping VUs for 30s over 6 stages (gracefulRampDown: 30s, gracefulStop: 30s)

===== Stress Test Summary =====

Total Requests: 658
Request Rate: 11.00 req/s
Average Duration: 16278.43 ms

===== Stress Test Summary =====

Total Requests: 658
Request Rate: 11.00 req/s
Average Duration: 16278.43 ms
95th Percentile: 31725.82 ms
Max Duration: 34927.70 ms
Failed Requests: 0.00%
```

```
1 import http from 'k6/http';
2 import { check, sleep } from 'k6';
3 import { Rate } from 'k6/metrics';
4
5 // Custom metrics
6 const errorRate = new Rate('errors');
7
8 // Stress test configuration
9 export const options = {
10   stages: [
11     { duration: '30s', target: 100 }, // Ramp up to 100 users
12     { duration: '1m', target: 200 }, // Ramp up to 200 users
13     { duration: '1m30s', target: 200 }, // Stay at 200 users (stress period)
14     { duration: '30s', target: 300 }, // Spike to 300 users
15     { duration: '1m', target: 300 }, // Stay at 300 users (breaking point)
16     { duration: '30s', target: 0 }, // Ramp down to 0 users
17   ],
18   thresholds: {
19     http_req_duration: ['p(95)<1000'], // 95% of requests should be below 1s
20     http_req_failed: ['rate<0.3'], // Failure rate should be below 30%
21   },
22};
```

```
const BASE_URL = __ENV.BASE_URL || 'http://localhost:5205';
const STOCK_CODES = ['2330', '2317', '2454', '2882', '2881', '2412', '2303', '2886', '1301', '2891'];

export default function () {
  const stockCode = STOCK_CODES[Math.floor(Math.random() * STOCK_CODES.length)];
  const userId = Math.floor(Math.random() * 1000) + 1;

  // Test 1: Create Order (write-heavy operation)
  const orderPayload = JSON.stringify({
    userId,
    stockCode,
    orderType: Math.random() > 0.5 ? 1 : 2, // Random buy/sell
    price: 500.0 + Math.random() * 100,
    quantity: (Math.floor(Math.random() * 10) + 1) * 1000, // 1000-10000
  });

  const orderResponse = http.post(`${BASE_URL}/api/v1/orders`, orderPayload, {
    headers: {
      'Content-Type': 'application/json',
      'Accept': 'application/json'
    },
  });

  const orderSuccess = check(orderResponse, {
    'order status is 201 or 400 or 404': (r) => [201, 400, 404].includes(r.status),
    'order response time < 3s': (r) => r.timings.duration < 3000,
  });

  if (!orderSuccess) {
    errorRate.add(1);
  } else {
    errorRate.add(0);
  }
}
```