讓我們用一種方式來開發吧

講者：Poy Chang



歡迎參加我們的每週四固定聚會



Poy Chang

任職於全美 100 大私人企業，負責企業內部 IT 解決方案設計與開發，專注於 .NET、Azure、Angular 等技術研究

- ✓ Microsoft MVP 微軟最有價值專家 2018 ~ Present
- ✓ STUDY4 / Azure Taiwan / Angular Taiwan
- ✓ 金融研訓院 / 資訊工業策進會 / 工業技術研究院 講師
- ✓ 2020 台北 .NET Conf 總召 / 講師
- ✓ 2019 台北 .NET Conf 總召
- ✓ 2019 廣州 Global Azure Bootcamp 講師
- ✓ 2019 台北 Insider Dev Tour 講師
- ✓ 2019 宜蘭 Angular TW Conf 講師
- ✓ 2018 台北 Azure Tech Day Party 講師



起源於 David 老師的一場演講

https://www.ghibli.jp 常識範圍使用

Framework Architecture

Infrastructure Layer (Cross-cutting concerns)

- Security
- Exception Handling
- Notification
- Logs
- Scheduling
- Others AOP

Presentation Layer

- Native Mobile App
- Web / Hybrid Mobile App
- Web-Based App
  - ASP.NET WebForm/MVC/SPA
- Desktop App (WPF/WinForm)
- HTML 5 + JavaScript framework(angularjs)

Services Layer

- Web Service Layer (WebAPI)
- Application Service Layer(optional)

Backend Application Service Layer

- Domain Components (BLL)
- Data Access Components (DAL / ORM)

.NET Framework 4.5.x (Cloud Compatible)

DB



這一切都是為了團隊開發
而做的架構設計

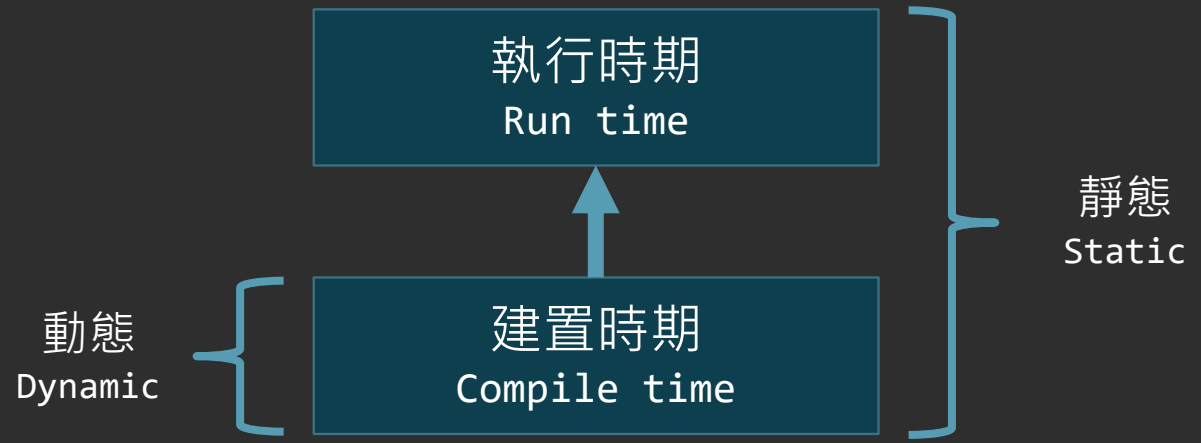
這一切都是為了快點開發
而做的速成設計

你有在用反射/動態嗎？

Reflection / Dynamic

Dynamic 動態型別

- C# 4.0 (.NET Framework 4.0) 開始提供
- 執行時期決議機制，只在執行期解析程式結構
  - dynamic 是靜態類型，但該物件會略過靜態類型檢查
  - 在基於 CLR 之上的 DLR 做執行時期繫結



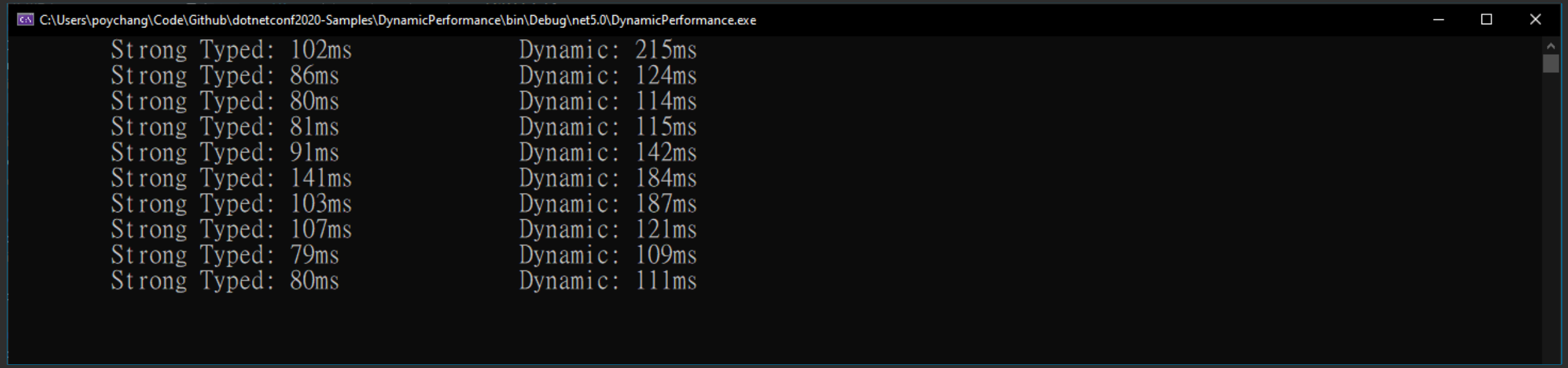
COM Interop

```
// 過去只能用強制轉型來處理
((Excel.Range)excelApp.Cells[1, 1]).Value2 = "Name";
Excel.Range range2008 = (Excel.Range)excelApp.Cells[1, 1];
```

```
// 有了 dynamic 之後，轉型會交由執行時期的 COM 繫結器處理
excelApp.Cells[1, 1].Value = "Name";
Excel.Range range2010 = excelApp.Cells[1, 1];
```

限制

- 動態物件無法使用擴充方法
- 動態型別無法取得靜態成員或建構式
- 無法直接指定成方法、匿名方法、Lambda，除非先轉型
- 效能是個軟肋



動手做

使用 dynamic 做 ASP.NET Core API 的資料繫結

專案程式碼 https://github.com/poychang/twMVC42



https://blog.poychang.net/using-dynamic-type-to-bind-data-in-asp-net-core/

The Common Language Runtime



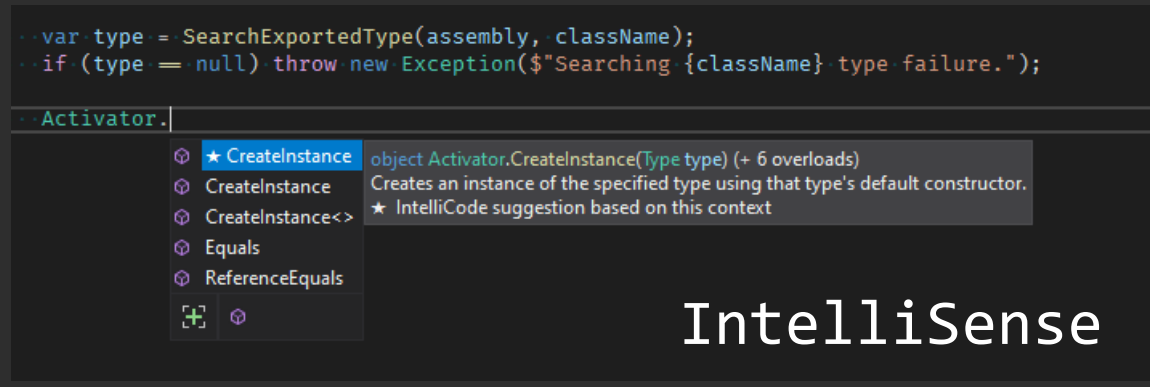
Reflection 反映

- .NET Framework 2.0 開始提供 System.Reflection 類別
- 組件（Assembly）包含至少一個模組（Module）
- 模組包含型別（Type），型別包含成員（Property, Method）



用在哪裡？

- 終極的延遲綁定
  - IDE 的 WinForm 控制項屬性視窗
  - Visual Studio 的 IntelliSense
- 即時生成/執行程式
  - 程式碼分析工具
  - 測試應用程式
- 讓使用者能夠自行選擇要執行的實作
- 可大幅簡化程式碼複雜度的時候



代價

- 反射機制的強大增加了開發者的靈活性
  - 針對特定問題提供強大解決方案
  - 對於基礎結構的程式碼非常有用
- 靈活性的代價
  - 效能損失
  - 安全性的限制（反射程式碼可能在某些環境無法使用）
  - 違反封裝原則
  - 程式碼變的更多，更難追蹤與理解

效能開銷

- 類別建構的開銷
  - 一次性的開銷，通常可以忽略
- 執行的開銷
  - 通常會比正常呼叫慢
  - 若應用程式大量使用 `Reflection` 做法，此問題更為彰顯

缺點

- 無法在編譯時期保證型別安全性
  - 嚴重依賴字串標識，所以喪失編譯時期的型別安全
- 執行速度相對較慢
  - 需先掃描執行程序的 Metadata，且會不停的執行字串搜尋
  - 字串搜尋過程中是執行不區分大小寫的比較

```
<Reference Include="Newtonsoft.Json, Version=12.0.0.0, Culture=neutral, PublicKeyToken=30ad4fe6b2a6aeed, processorArchitecture=MSIL">
  <HintPath>..\packages\Newtonsoft.Json.12.0.3\lib\net45\Newtonsoft.Json.dll</HintPath>
</Reference>
<Reference Include="System" />
<Reference Include="System.Buffers, Version=4.0.3.0, Culture=neutral, PublicKeyToken=cc7b13ffcd2ddd51, processorArchitecture=MSIL">
  <HintPath>..\packages\System.Buffers.4.5.1\lib\net461\System.Buffers.dll</HintPath>
</Reference>
```

載入外部 DLL

- System.Reflection
  - Assembly.Load()
  - Assembly.LoadFrom()
    - 可指定特定 DLL 檔案，並載入相依的 DLL
    - 若曾經載入相同標識的 DLL，直接回傳已載入的物件
  - Assembly.LoadFile()
    - 只載入指定的 DLL 檔案
    - 可以將相同標幟的 DLL 多次載入到 AppDomain 中

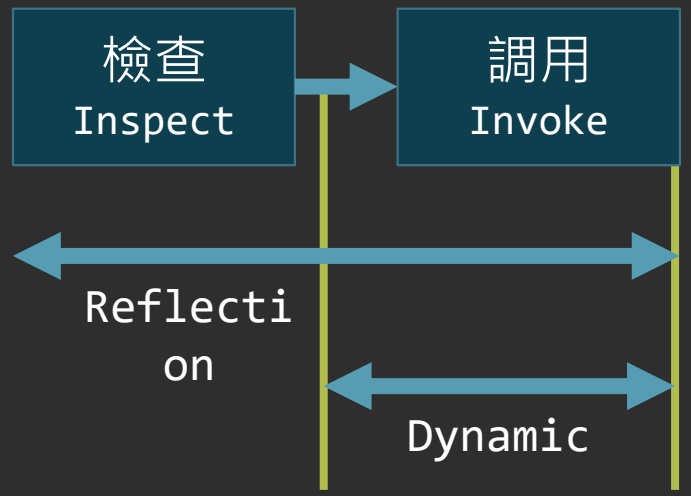
建議每次建置產出 DLL 時都更改版本號，使之具有唯一標識，確保行為符合預期

3 種實體化 API

- System.Activator
  - CreateInstance()
  - CreateInstanceFrom()
- System.AppDomain
  - CreateInstance()
  - CreateInstanceAndUnwrap()
  - CreateInstanceFrom()
  - CreateInstanceFromAndUnwrap()
- System.Reflection.ConstructorInfo
  - Invoke()

從已載入的型別中
實體化物件

Reflection 和 Dynamic 的差異



	Reflection	Dynamic
Inspect (metadata)	Yes	
Invoke public member	Yes	Yes
Invoke private member	Yes	
Caching		Yes
Static class	Yes	

Dynamic

Reflection

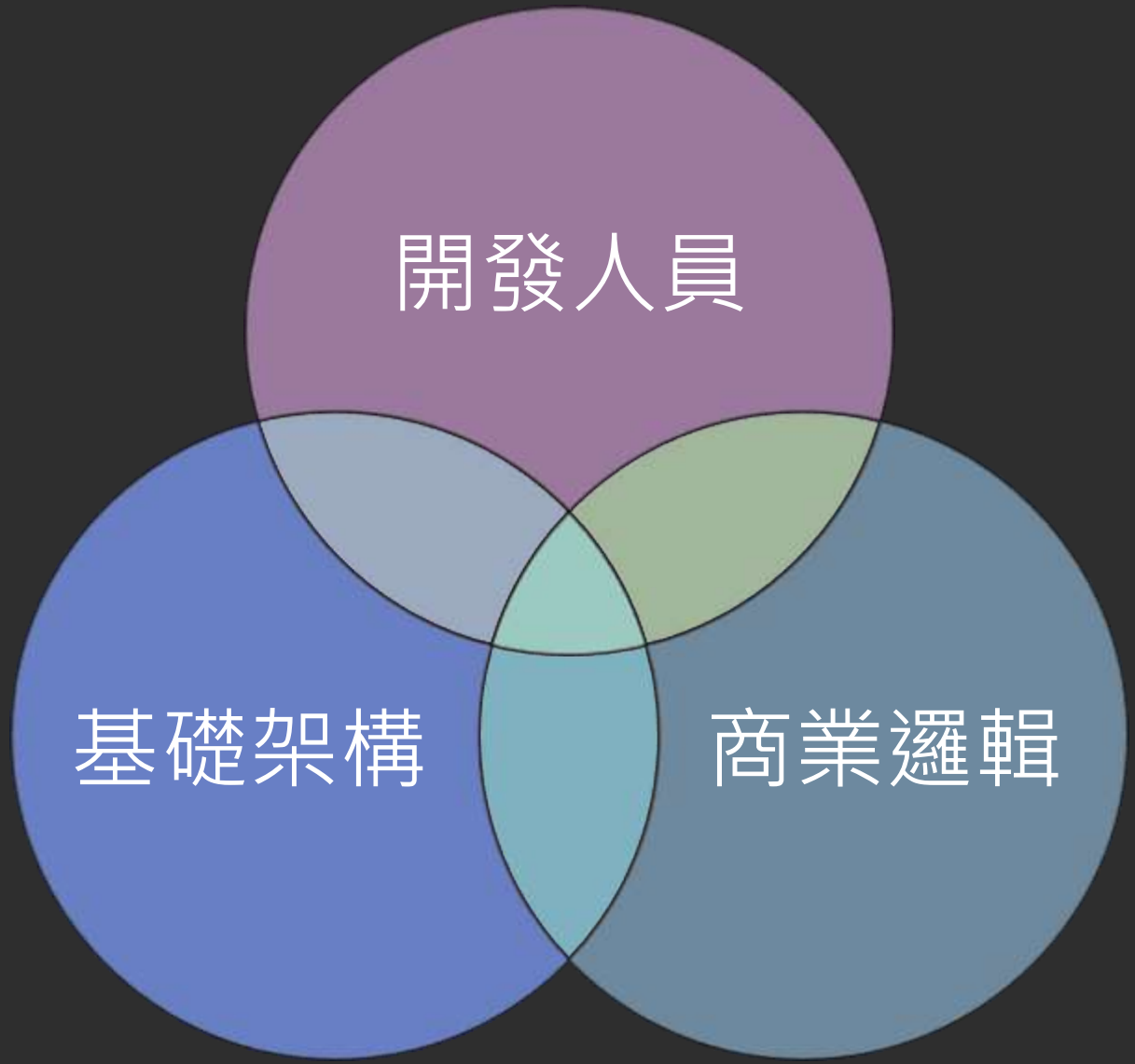
Delegate

Expression

KingKong Bruce記事: Reflection-使用反射執行方法的7種方式

https://blog.kkbruce.net/2017/01/reflection-method-invoke-7-ways.html

架構設計的思考點



關鍵的設計原則

- ✓ 關注點分離
- ✓ 單一責任原則
- ✓ 最少知識原理
- ✓ 不要重複自己
- ✓ 最小化前期設計風險

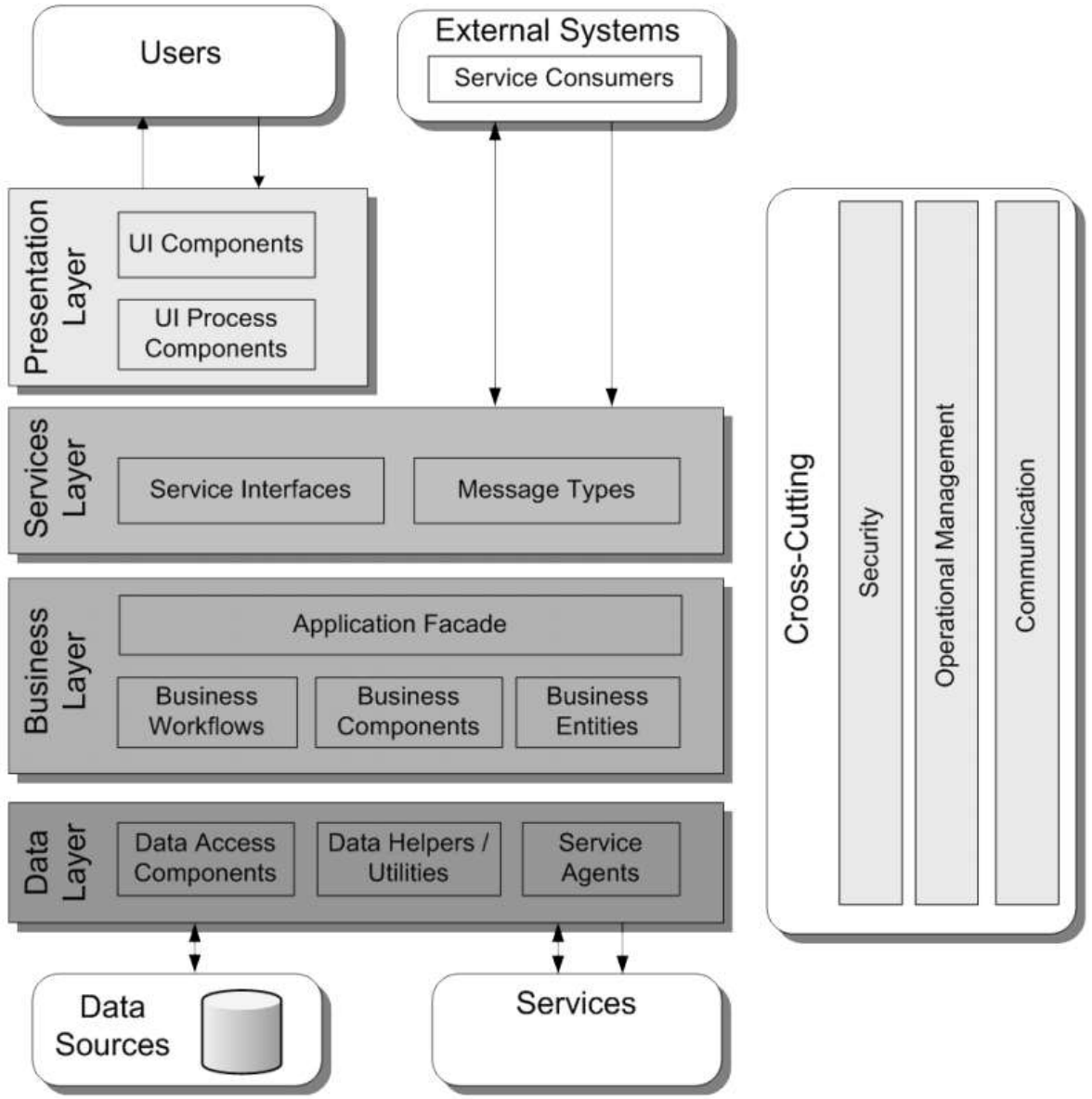
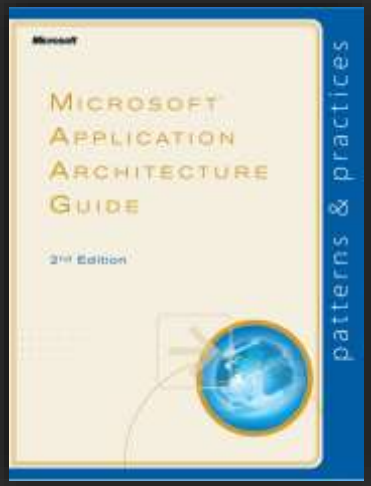
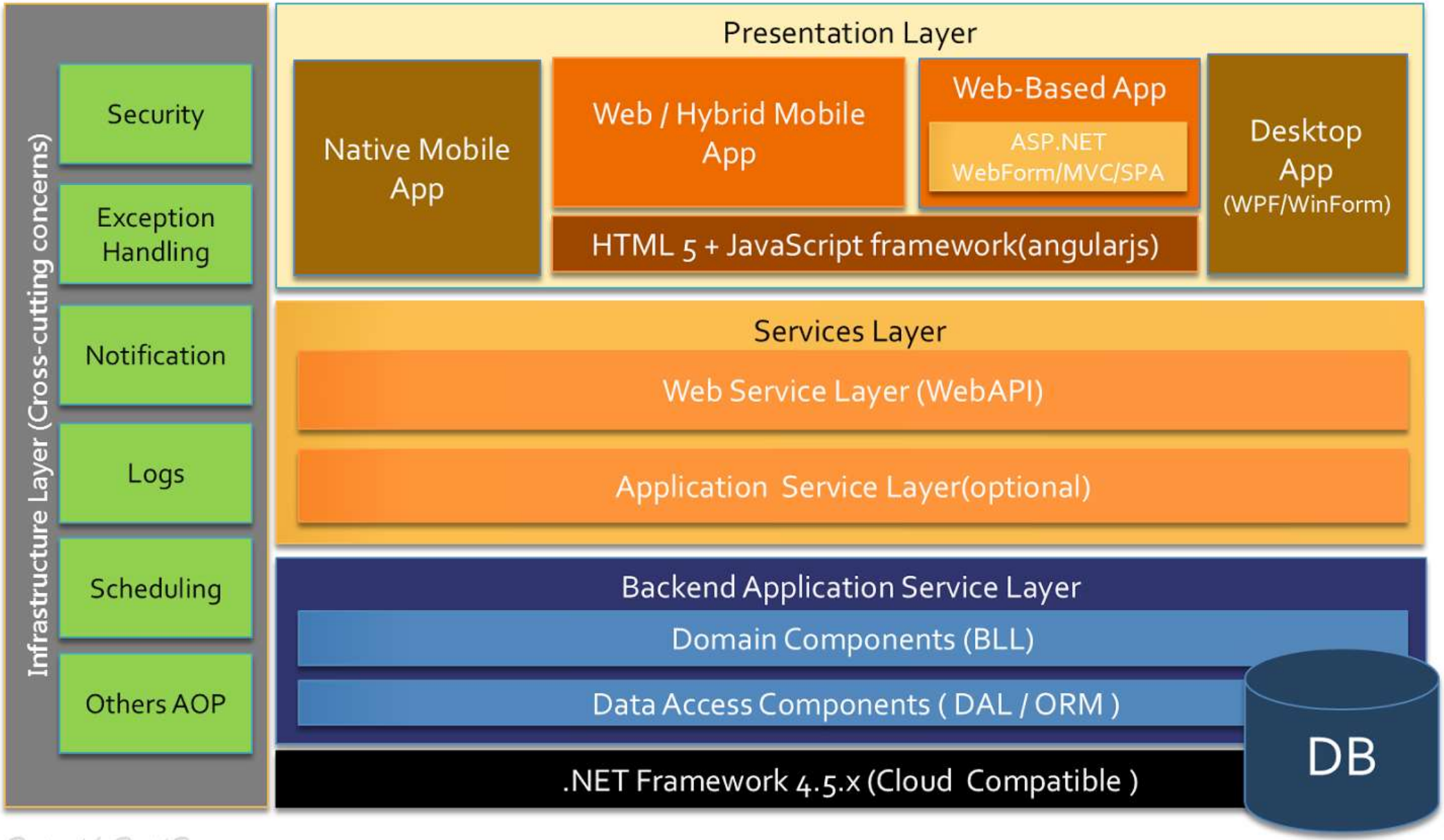


Figure 1 Common application architecture

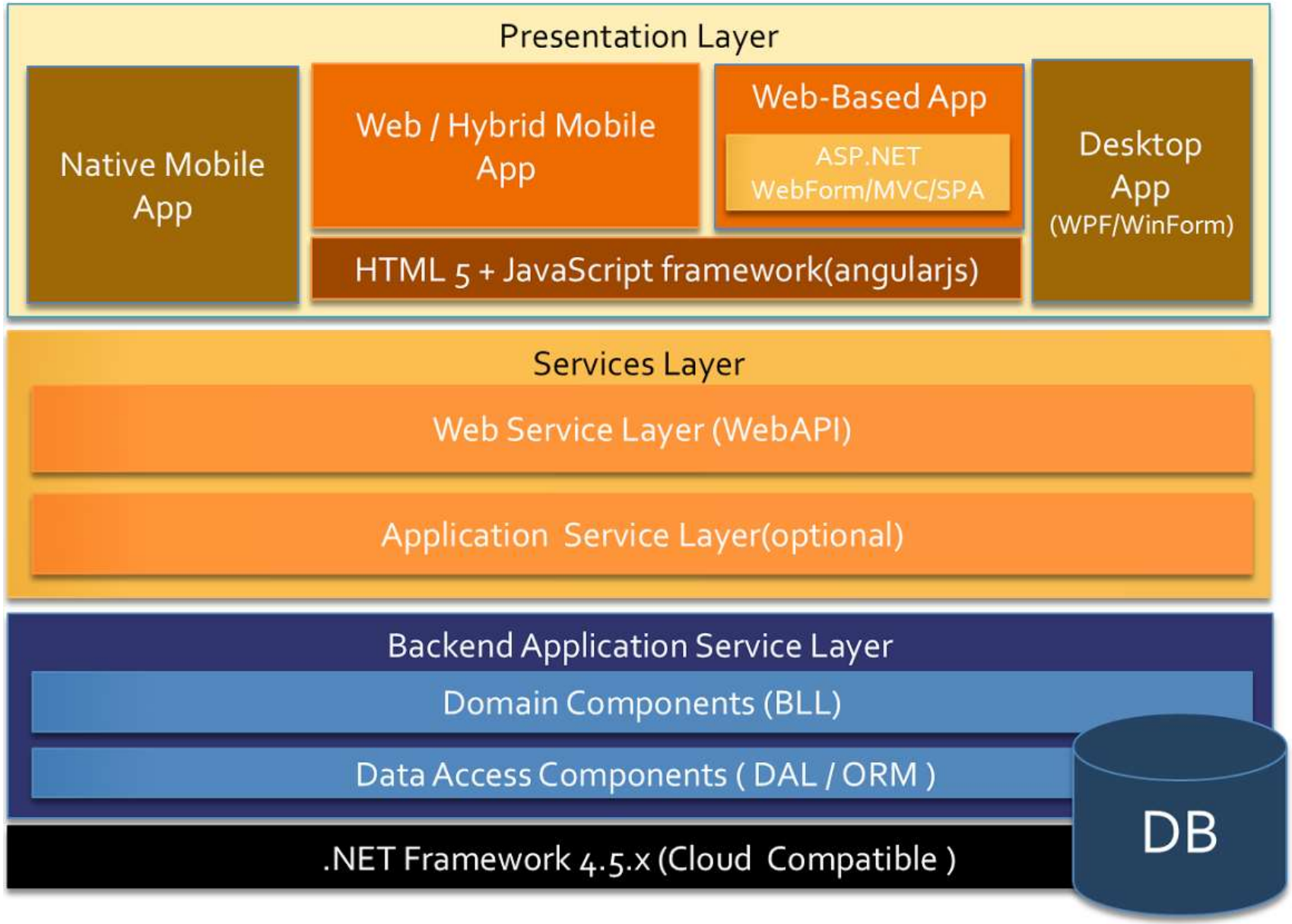


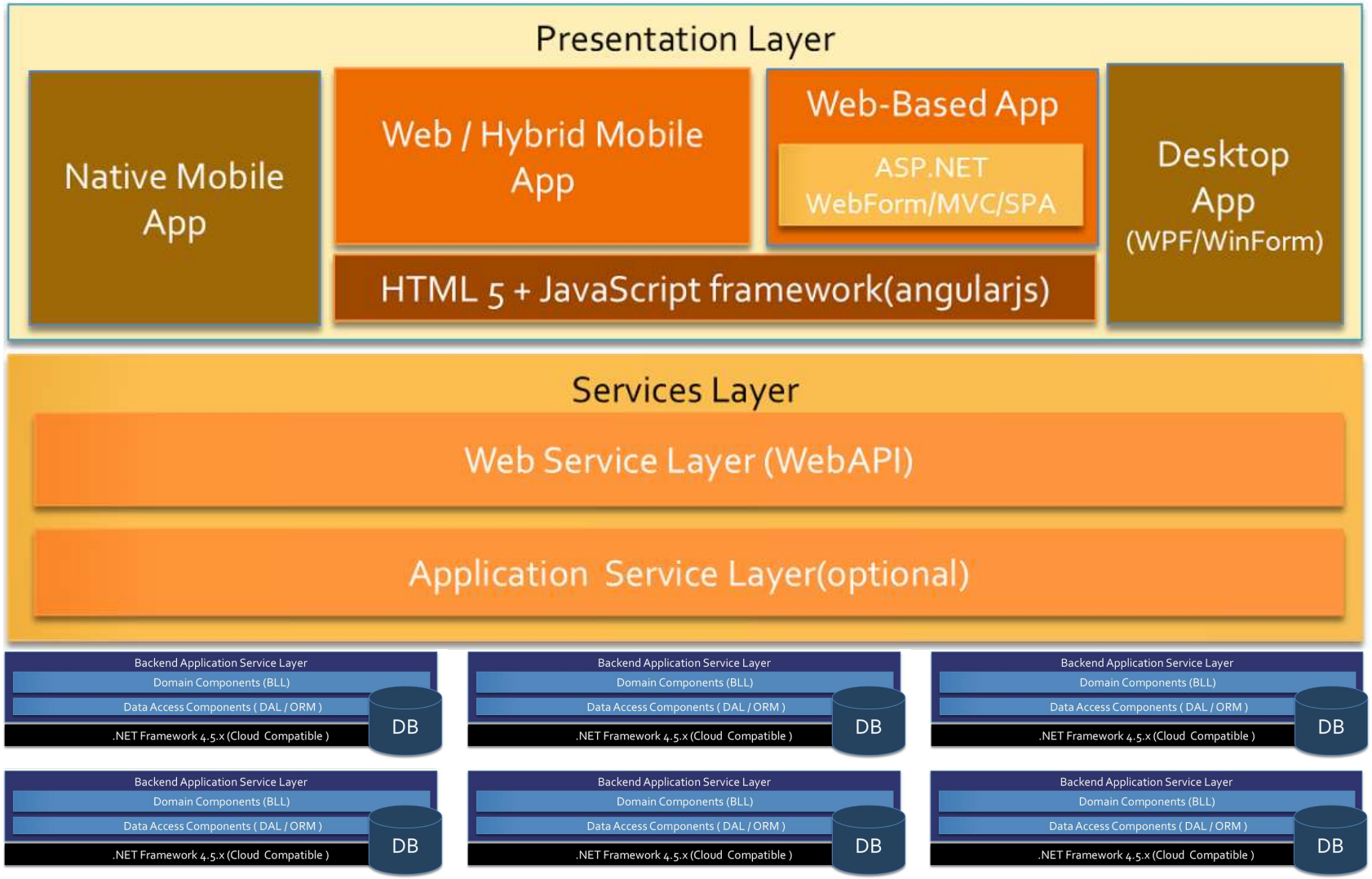
Microsoft Application Architecture Guide

Framework Architecture

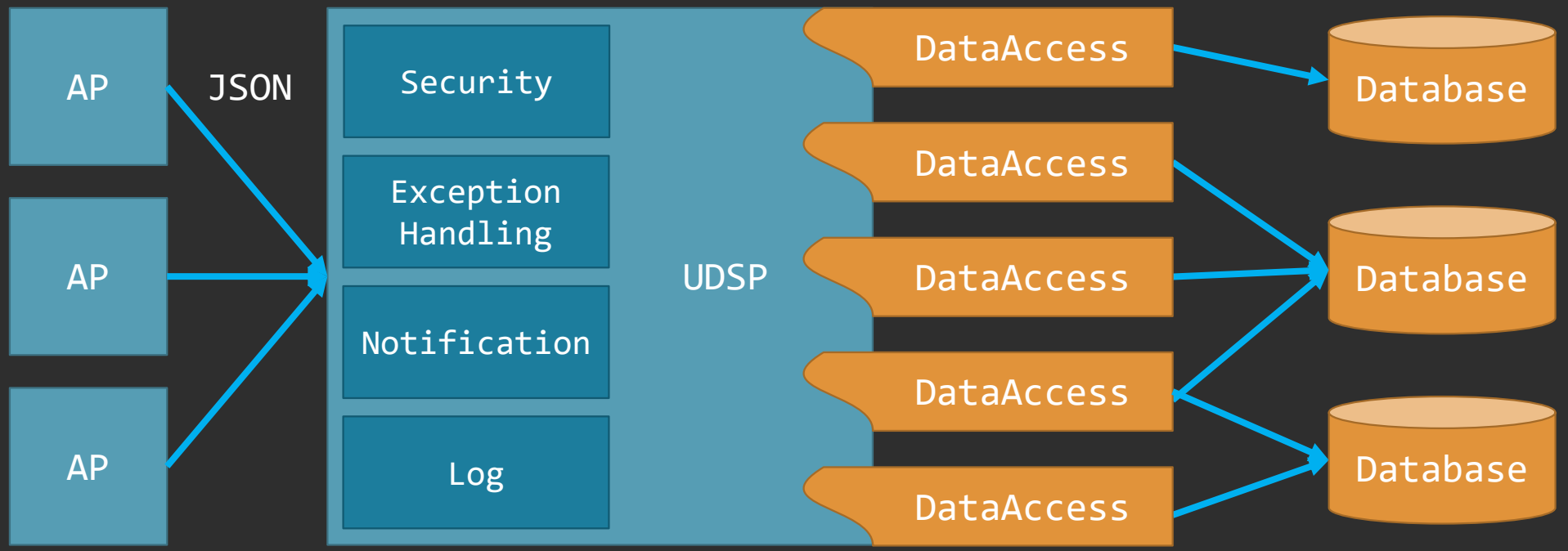


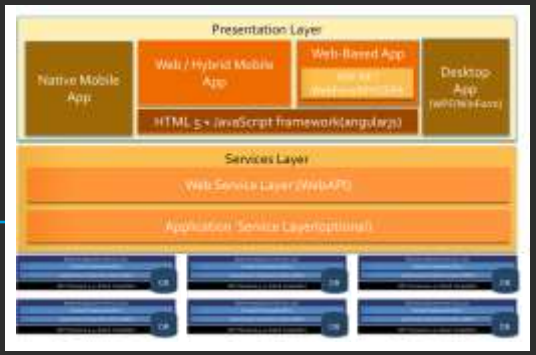
Designed by David Dong


Presentation Layer
Native Mobile App
Web / Hybrid Mobile App
Web-Based App
ASP.NET WebForm/MVC/SPA
Desktop App (WPF/WinForm)
HTML 5 + JavaScript framework(angularjs)
Services Layer
Web Service Layer (WebAPI)
Application Service Layer(optional)
Backend Application Service Layer
Domain Components (BLL)
Data Access Components (DAL / ORM)
.NET Framework 4.5.x (Cloud Compatible)
DB


Presentation Layer
Native Mobile App
Web / Hybrid Mobile App
Web-Based App
ASP.NET WebForm/MVC/SPA
Desktop App (WPF/WinForm)
HTML 5 + JavaScript framework(angularjs)
Services Layer
Web Service Layer (WebAPI)
Application Service Layer(optional)
Backend Application Service Layer
Domain Components (BLL)
Data Access Components (DAL / ORM)
.NET Framework 4.5.x (Cloud Compatible)
DB

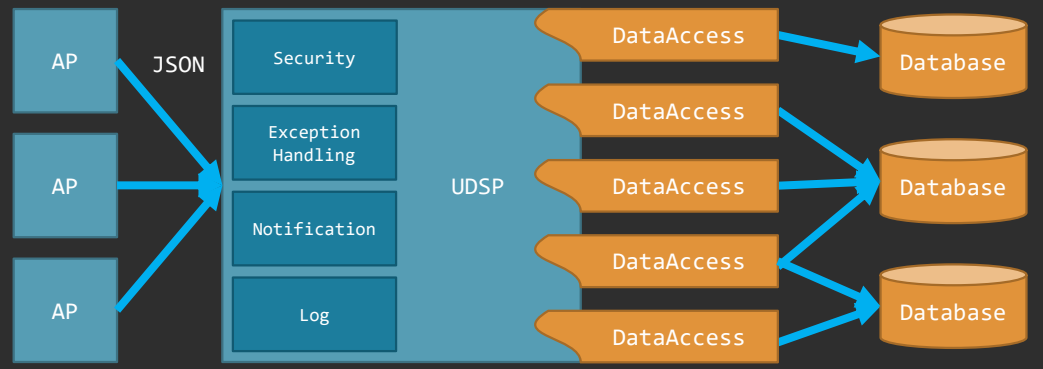
Universal Data Platform



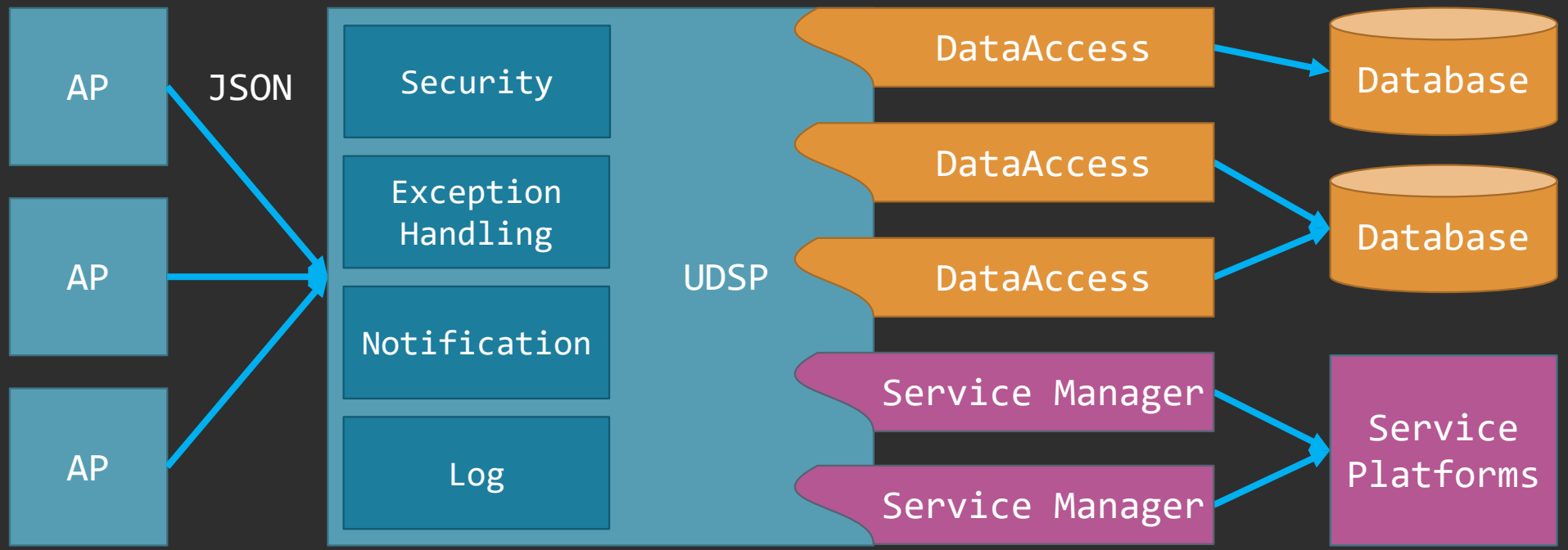


Universal Data Platform - JSON Payload

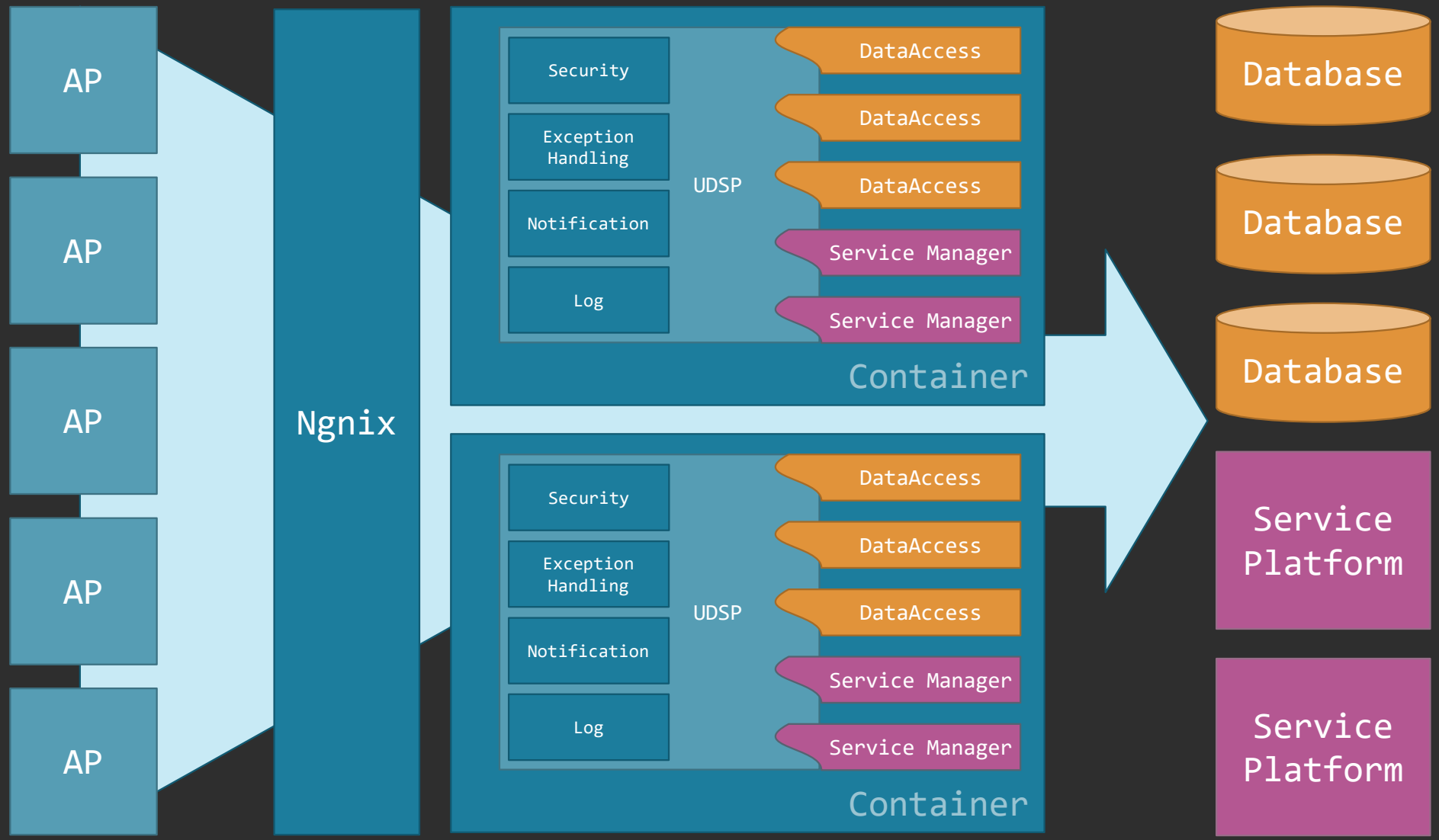
```
{
  "IsDao":          true,
  "DbName":         "B2BDB",
  "ProjectName":    "B2B",
  "ContractName":   "B2B.Contract",
  "ContractType":   "B2BDataService",
  "DllName":        "B2B.DataAccess",
  "ClassName":      "DemoDal",
  "MethodName":     "ReadDemoReport",
  "JsonParam":      "{\"Version\":\"20210320\"}"
}
```



Universal Data & Service Platform



Universal Data & Service Platform



Universal Data & Service Platform

UDSP 提供統一的呼叫服務方法
那 Client 端有統一地開發方法嗎？

直接來看 Code

使用泛型主機
開發應用程式

專案程式碼 https://github.com/poychang/twMVC42



thanks for your attention!

blog.poychang.net

www.facebook.com/poychang.tech

Blog 是記錄知識的最佳平台



https://dotblogs.com.tw

SkillTree 為了確保內容與實務不會脫節，我們都是聘請企業顧問等級並且目前依然在職場的業界講師，我們不把時間浪費在述說歷史與沿革，我們並不是教您考取證照，而是教您如何上場殺敵，拳拳到肉的內容才是您花錢想要聽到的，而這也剛好是我們擅長的。



https://skilltree.my

天瓏資訊圖書

