使用Unity實現.NET相依性注入架構設計

陳葵懋 Ian Chen
http://codeian.idv.tw/

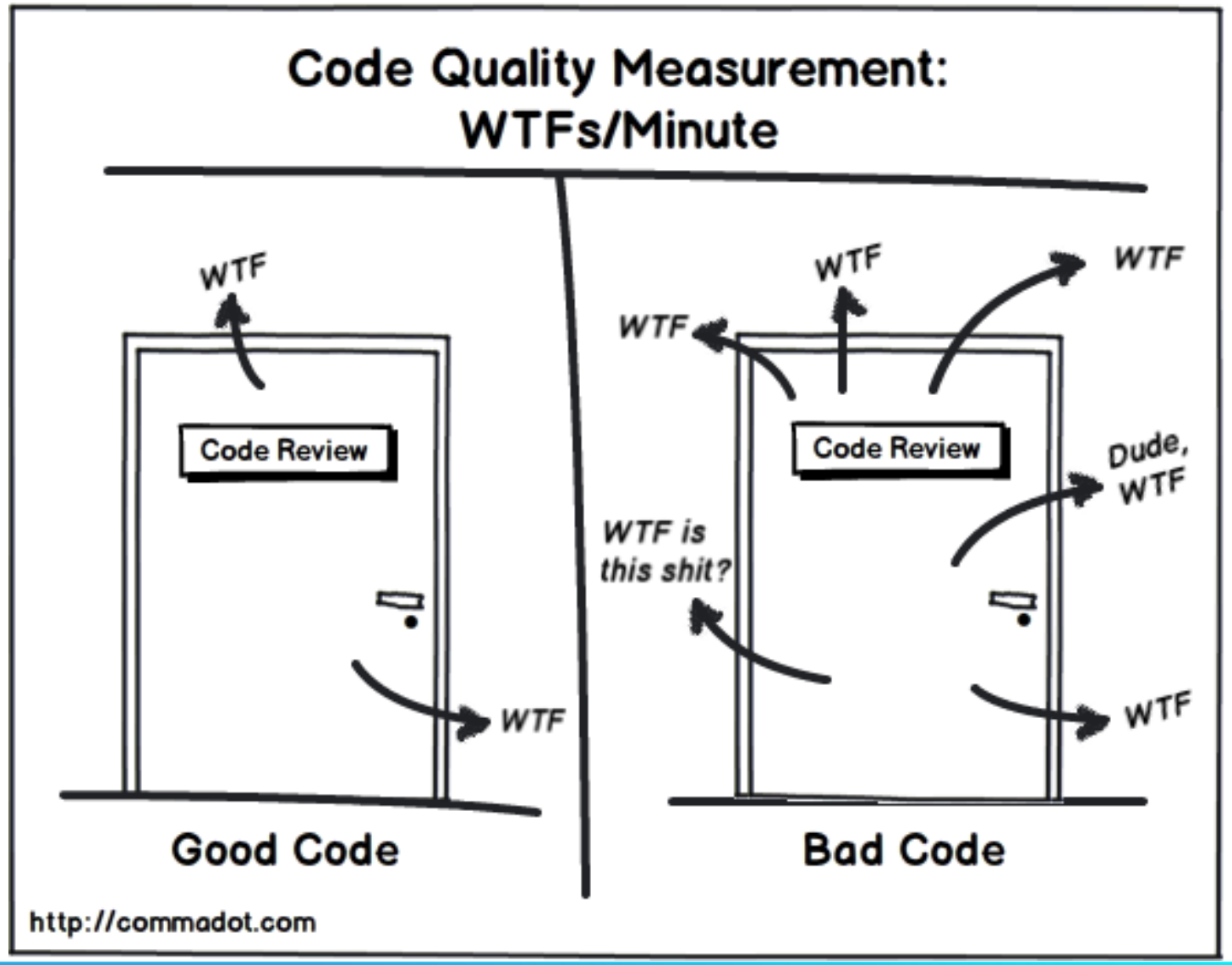
codeian.idv.tw

- Microsoft MVP
- TechDays Taiwan 2014 / 2015講師
- 2015 微軟實戰課程日講師
- Channe 9線上課程講師
- K.NET & Study4.TW社群講師
- HTML5 & JavaScript 程式開發實戰書籍共同作者
- Microsoft MSDN 技術文章作者

Agenda

- 為什麼要DI
- 重構無所不在
- Use Unity
- Q&A

今天不談 Design pattern


Code Quality Measurement:
WTFs/Minute
WTF
Code Review
WTF
Good Code
WTF
WTF
WTF
Code Review
Dude,
WTF
WTF is
this shit?
WTF
Bad Code
http://commadot.com

為什麼要DI

- 可維護性
  - 程式修改時所需要花費的時間成本少
- 寬鬆耦合
  - 類別間的相依性低

OOP：單一職責
The Single Responsibility Principle

單一職責

- 每個Class / Function只負責一件事情職責的內容
- 處理不同的事務應該透過組裝來實現

```
0 個參考
public class Course
{
    0 個參考
    public String Name { get; set; }
    0 個參考
    public DateTime DateInfo { get; set; }
    0 個參考
    public String TeacherName { get; set; }
    0 個參考
    public String Gender { get; set; }
}
```

```
0 個參考
public class Course
{
    0 個參考
    public String Name { get; set; }
    0 個參考
    public DateTime DateInfo { get; set; }
    0 個參考
    public String TeacherName { get; set; }
    0 個參考
    public String Gender { get; set; }
}
```

```
0 個參考
public class Course
{
    0 個參考
    public String Name { get; set; }
    0 個參考
    public DateTime DateInfo { get; set; }
    0 個參考
    public String TeacherName { get; set; }
    0 個參考
    public String Gender { get; set; }
}
```

Issue：多個職責

課程資訊+講師資訊

```
0 個參考
public class Course
{
    0 個參考
    public String Name { get; set; }
    0 個參考
    public DateTime DateInfo { get; set; }
    0 個參考
    public Teacher CourseTeacher { get; set; }
}


1 個參考
public class Teacher
{
    0 個參考
    public String TeacherName { get; set; }
    0 個參考
    public String Gender { get; set; }
}
```

真實WTF的案（暗）例

這是一個計算薪資的類別

單一類別高達18475行的程式碼

```
810        protected override void RenderChildren(System.Web.UI.HtmlTextW
818
819        Property
1897
1898       Methods
15791
15792      Events
18385
18386
18391      private List<UsageAmountTable> DtConvert(DataTable dt)...
18415
18416      private void WriteEventLog(string ex)...
18422    }
18423
18424
18427    public class ResultInfo...
18474  }
18475
```

Issue

- 太多全域共用的變數
- 類別負責的職責太多
- 可維護性低

需求抽象化

需求抽象化

- 計算薪資
- 需要知道怎麼計算基本薪資
- 需要知道怎麼計算請假扣款
- 需要知道怎麼計算加班費

需求抽象化

- 計算薪資
- ~~需要知道怎麼計算基本薪資~~
- ~~需要知道怎麼計算請假扣款~~
- ~~需要知道怎麼計算加班費~~

需求抽象化

- 我要計算薪資
- 我要計算基本薪
- 我要計算加班費
- 我要計算請假扣款

重構第一步：單一職責

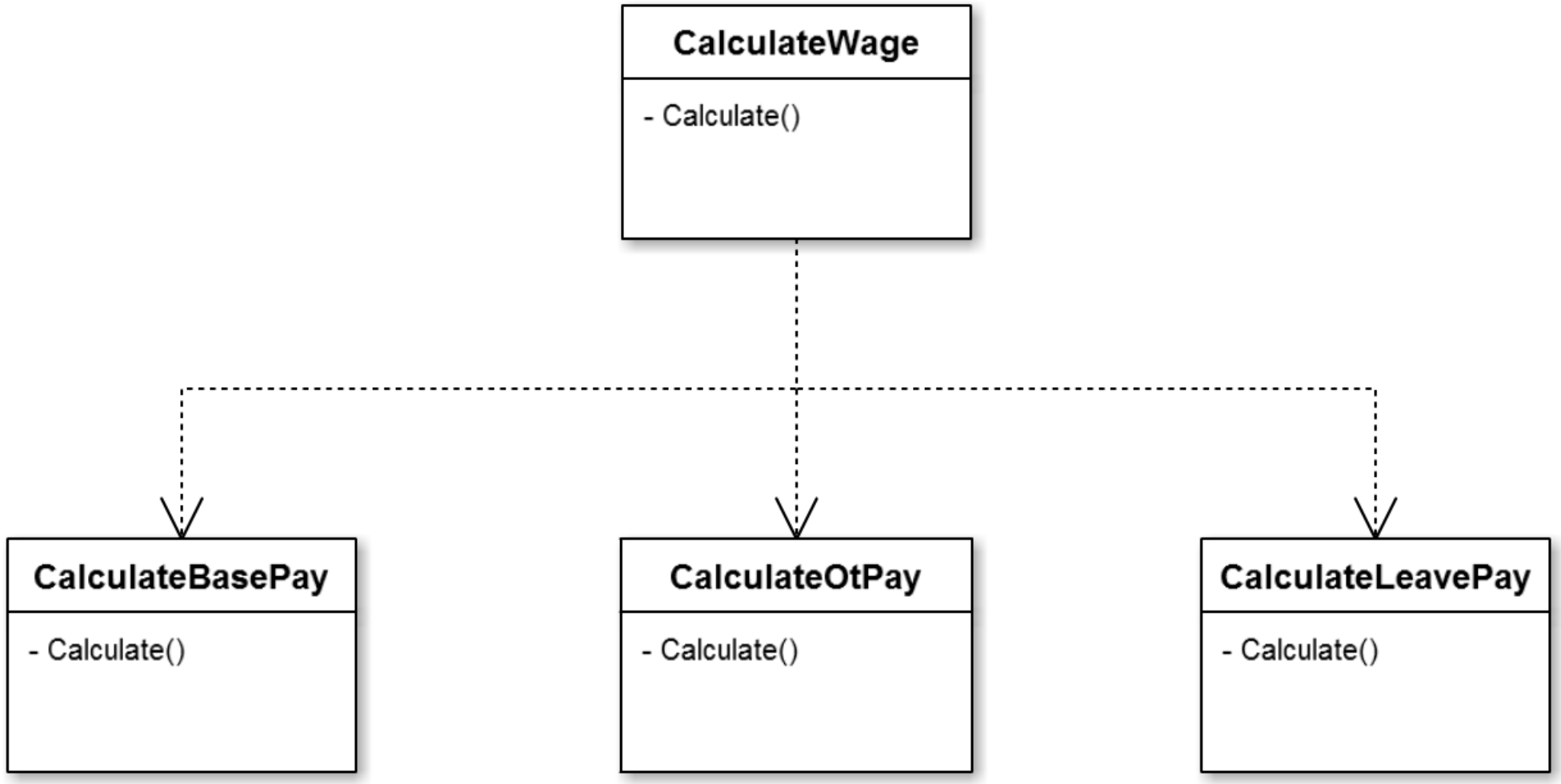
依單一職責的概念，拆解成多個小類別

```
/// <summary> 基本薪
0 個參考
public class CalculateBasePay
{
    0 個參考
    public int Calculate()...
}

/// <summary> 加班費
0 個參考
public class CalculateOtPay
{
    0 個參考
    public int Calculate()...
}

/// <summary> 請假扣款
0 個參考
public class CalculateLeavePay
{
    0 個參考
    public int Calculate()...
}
```

```
0 個參考
public class CalculateWage
{
    0 個參考
    public int Calculate()
    {
        int result = 0;
        var basepay = new CalculateBasePay();
        var otpay = new CalculateBasePay();
        var leavepay = new CalculateBasePay();
        result += basepay.Calculate();
        result += otpay.Calculate();
        result += leavepay.Calculate();
        return result;
    }
}
```

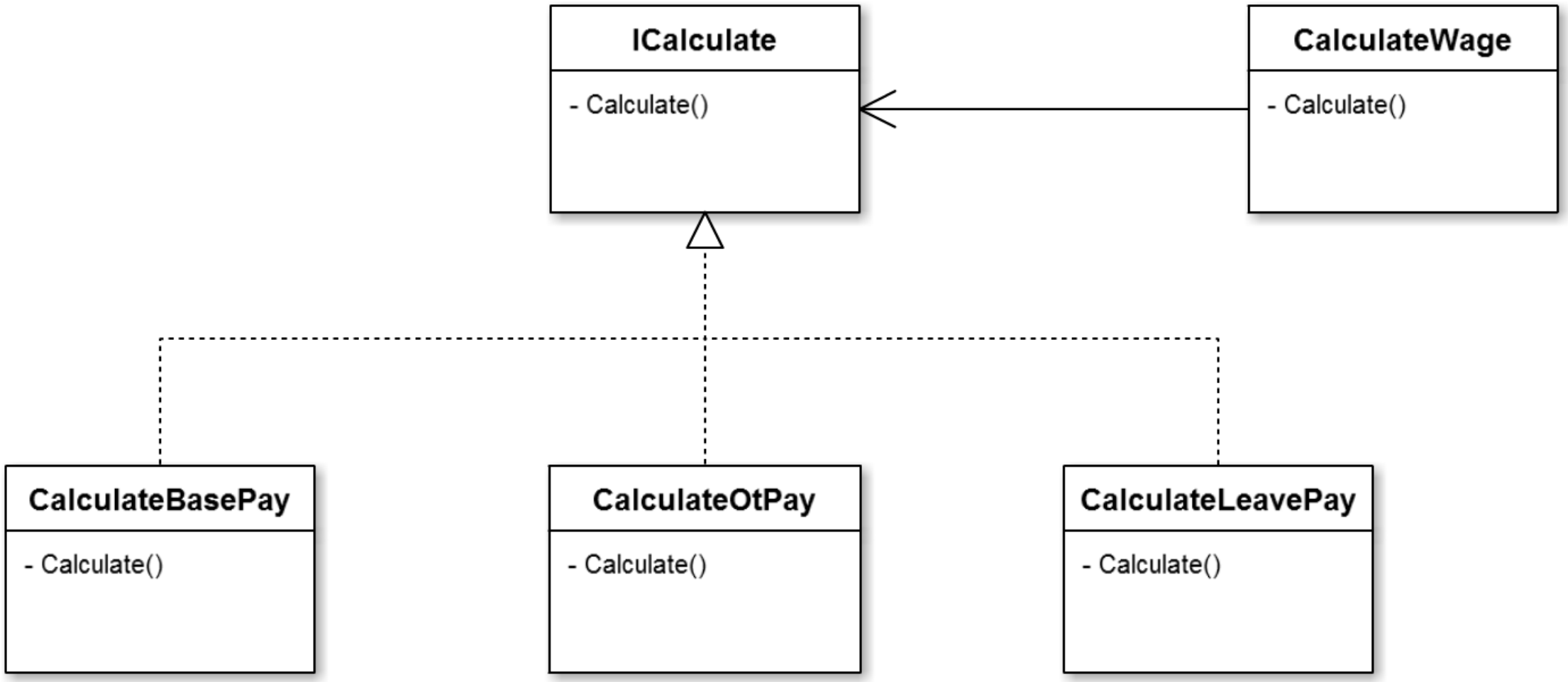
```
0 個參考
public class CalculateWage
{
    0 個參考
    public int Calculate()
    {
        int result = 0;
        var basepay = new CalculateBasePay();
        var otpay = new CalculateBasePay();
        var leavepay = new CalculateBasePay();
        result += basepay.Calculate();
        result += otpay.Calculate();
        result += leavepay.Calculate();
        return result;
    }
}
```

Issue：類別間耦合度高



重構第二步：介面隔離

介面隔離


ICalculate
- Calculate()
CalculateWage
- Calculate()
CalculateBasePay
- Calculate()
CalculateOtPay
- Calculate()
CalculateLeavePay
- Calculate()

相依於介面

```
public class CalculateWage
{
    2 個參考
    public ICalculate BasePay { get; set; }
    2 個參考
    public ICalculate OtPay { get; set; }
    2 個參考
    public ICalculate LeavePay { get; set; }

    1 個參考
    public int Calculate()
    {
        int result = 0;
        result += BasePay.Calculate();
        result += OtPay.Calculate();
        result += LeavePay.Calculate();
        return result;
    }
}
```

AP端組裝介面實作

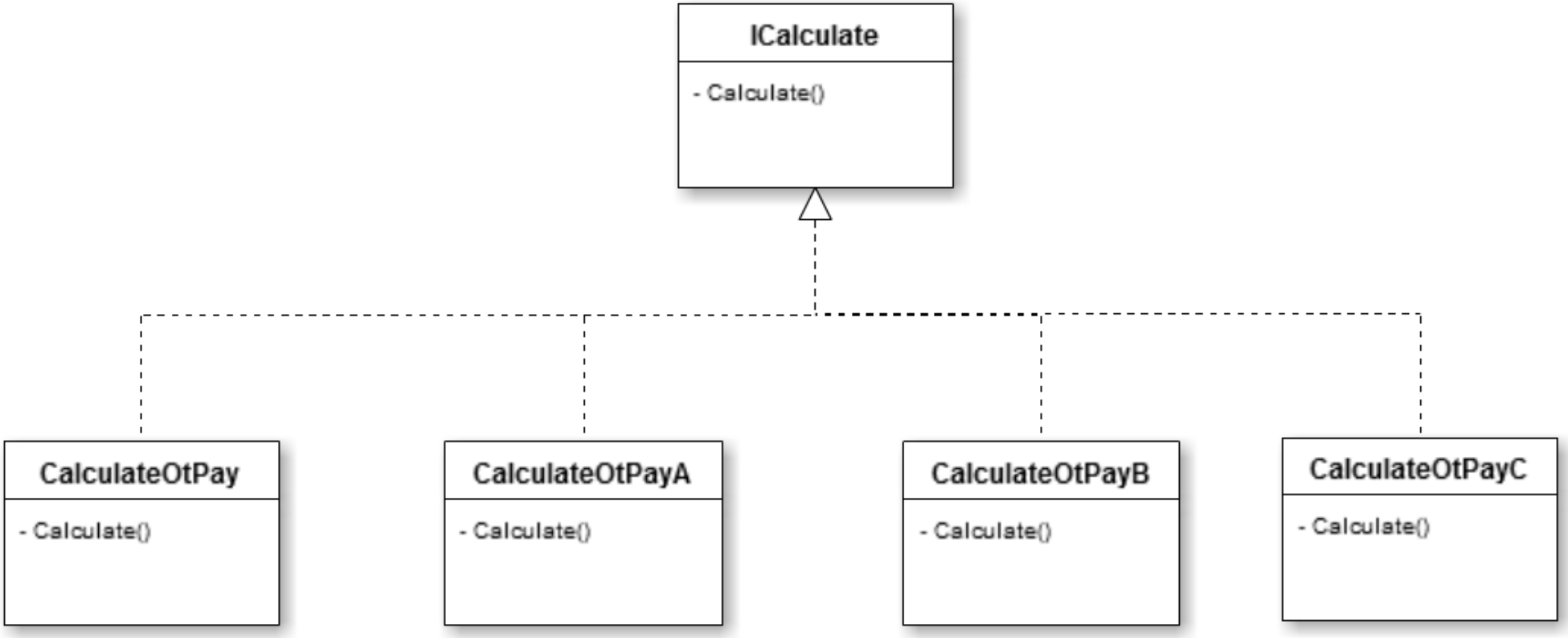
```
var basepay = new CalculateBasePay();
var leavepay = new CalculateLeavePay();
var otpay = new CalculateOtPay();

var wagecal = new CalculateWage();
wagecal.BasePay = basepay;
wagecal.OtPay = otpay;
wagecal.LeavePay = leavepay;
int result = wagecal.Calculate();
```

A客戶：加班費一律2倍計算

B客戶：加班費依勞基法計算

C客戶：加班費平日2倍計算，假日2.5倍計算

實作介面


ICalculate
- Calculate()
CalculateOtPay
- Calculate()
CalculateOtPayA
- Calculate()
CalculateOtPayB
- Calculate()
CalculateOtPayC
- Calculate()

```
switch (cus)
{
    case "A":
        otpay = new CalculateOtPayA();
        break;
    case "B":
        otpay = new CalculateOtPayB();
        break;
    case "C":
        otpay = new CalculateOtPayC();
        break;
    default:
        otpay = new CalculateOtPay();
        break;

}

var wagecal = new CalculateWage();
wagecal.BasePay = basepay;
wagecal.OtPay = otpay;
wagecal.LeavePay = leavepay;
int result = wagecal.Calculate();
```

```
switch (cus)
{
    case "A":
        otpay = new CalculateOtPayA();
        break;
    case "B":
        otpay = new CalculateOtPayB();
        break;
    case "C":
        otpay = new CalculateOtPayC();
        break;
    default:
        otpay = new CalculateOtPay();
        break;

}
```

```
var wagecal = new CalculateWage();
wagecal.BasePay = basepay;
wagecal.OtPay = otpay;
wagecal.LeavePay = leavepay;
int result = wagecal.Calculate();
```

Add New Cus D case “D”
Add New Cus E case “E”
Add New Cus F case “F”
Add New Cus G case “G”
…
…
…

重構第三步：DI 上場

Use Unity

- 執行時期(runtime)才決定實作類別
- 支援設定檔動態抽換

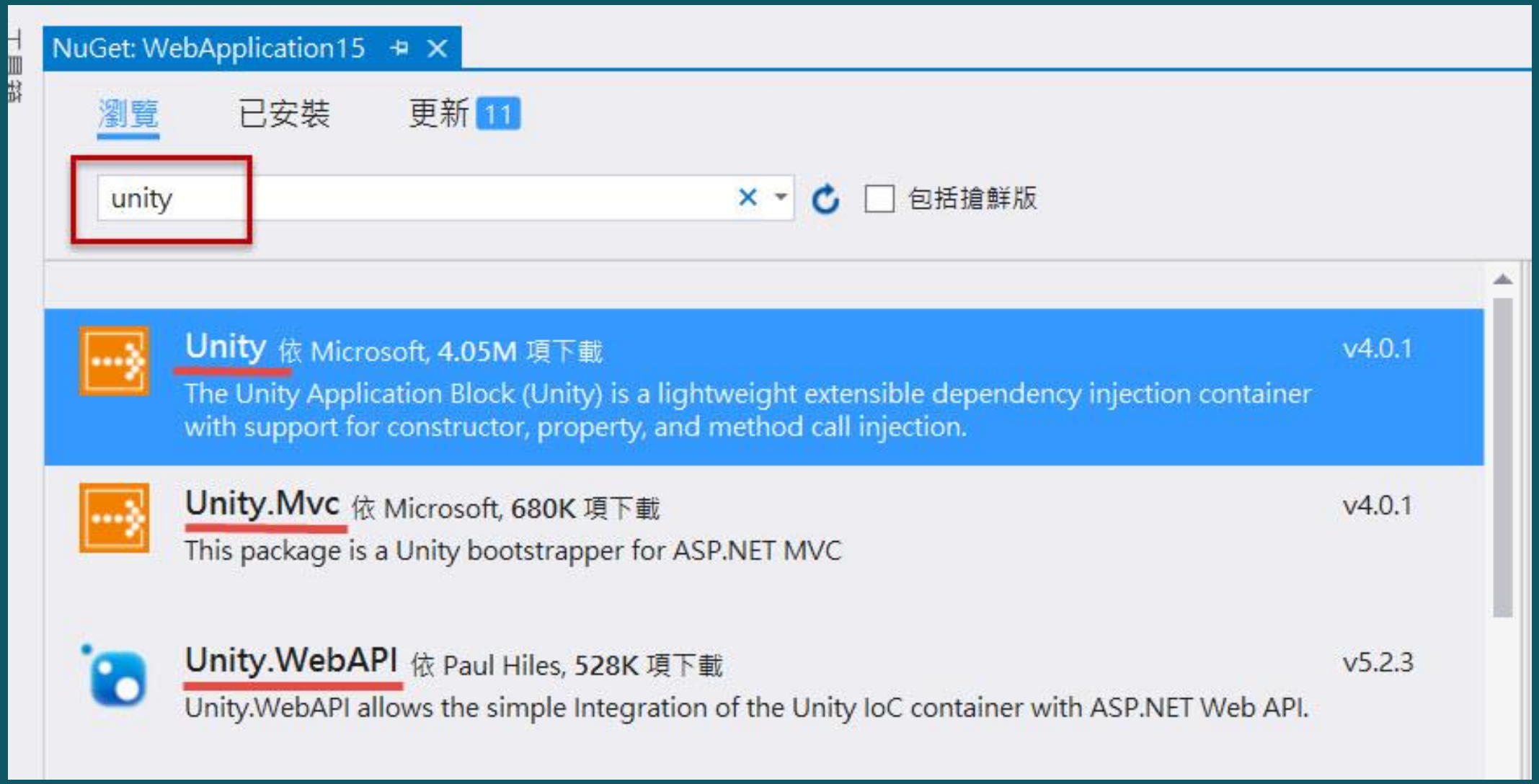
常見的 DI 容器

- Autofac
- Spring.NET
- Unity
- ……

Unity

- 由微軟的 Patterns & Practices group 所開發
- 獨立的類別庫
- 支援WebAPI / ASP.NET MVC / Web form / windows application

Use Unity


NuGet: WebApplication15
瀏覽
已安裝
更新 11
unity
包括搶鮮版
Unity 依 Microsoft, 4.05M 項下載
v4.0.1
The Unity Application Block (Unity) is a lightweight extensible dependency injection container with support for constructor, property, and method call injection.
Unity.Mvc 依 Microsoft, 680K 項下載
v4.0.1
This package is a Unity bootstrapper for ASP.NET MVC
Unity.WebAPI 依 Paul Hiles, 528K 項下載
v5.2.3
Unity.WebAPI allows the simple Integration of the Unity IoC container with ASP.NET Web API.

Use Unity

- Microsoft.Owin.Security.OAuth
- Microsoft.Owin.Security.Twitter
- Microsoft.Practices.ServiceLocation
- Microsoft.Practices.Unity
- Microsoft.Practices.Unity.Configuration
- Microsoft.Practices.Unity.RegistrationByConvention
- Microsoft.ScriptManager.MSAjax
- Microsoft.ScriptManager.WebForms

Use Unity

- 建立設定檔
  - 只要修改檔案內容就可以改變用應用程式的實作邏輯
  - 不須重新編輯程式或是類別庫

Use Unity

```
<configuration>
  <configSections>
    <!-- IOC unity -->
    <section name="unity"
    type="Microsoft.Practices.Unity.Configuration.UnityConfigurationSection
    , Microsoft.Practices.Unity.Configuration"/>
```

Use Unity

```xml
<!-- IOC unity -->
<unity xmlns="http://schemas.microsoft.com/practices/2010/unity">
  <container>
    <register
     type="WebApplication15.Models.ICalculate, WebApplication15"
     mapTo="WebApplication15.Models.CalculateBasePay, WebApplication15"
     name="CalBasePay" />

    <register
     type="WebApplication15.Models.ICalculate, WebApplication15"
     mapTo="WebApplication15.Models.CalculateOtPay, WebApplication15"
     name="CalOtPay" />

    <register
     type="WebApplication15.Models.ICalculate, WebApplication15"
     mapTo="WebApplication15.Models.CalculateLeavePay, WebApplication15"
     name="CalLeavePay" />
  </container>
</unity>
</configuration>
```

type = “ 抽象類別或介面名稱，組件名稱 ”

mapTo = “ 實作的類別名稱，組件名稱 ”

name = “ 別名 ”

多個類別實作同一介面，需要指定 name

Use Unity

- type = " 抽象類別或介面名稱 , 組件名稱 "
- mapTo = " 實作的類別名稱 , 組件名稱 "
- name = " 別名 "
  - 多個類別實作同一介面，需要指定name

```
<register
 type="WebApplication15.Models.ICalculate, WebApplication15"
 mapTo="WebApplication15.Models.CalculateOtPay, WebApplication15"
 name="CalOtPay" />
```

Use Unity

- GAC (assembly)設定檔的差異
  - 指定GAC
  - 拿掉register type 指定的dll name

```
<!-- IOC unity -->
<unity xmlns="http://schemas.microsoft.com/practices/2010/unity">
  <assembly name="Mylibrary.Framework, Version=4.0.0.0, Culture=neutral, PublicKeyToken=xxxxxxxxxxxxx" />
  <container>
    <register
     type="WebApplication15.Models.ICalculate"
     mapTo="WebApplication15.Models.CalculateBasePay, WebApplication15"
     name="CalBasePay" />
```

Use Unity

```
namespace WebApplication15.Models
{
    2 個參考
    public static class CalculateOtFactory
    {
        2 個參考
        public static ICalculate CalOtPay { get; private set; }

        0 個參考
        static CalculateOtFactory()
        {
            IUnityContainer container = new UnityContainer().LoadConfiguration();

            if (container.IsRegistered<ICalculate>("CalOtPay"))
            {
                CalOtPay = container.Resolve<ICalculate>("CalOtPay");
            }
        }
    }
}
```

Use Unity

```
ICalculate otpay;

switch (cus)
{
    case "A":
        otpay = new CalculateOtPayA();
        break;
    case "B":
        otpay = new CalculateOtPayB();
        break;
    case "C":
        otpay = new CalculateOtPayC();
        break;
    default:
        otpay = new CalculateOtPay();
        break;

}
```

```
var wagecal = new CalculateWage();
wagecal.BasePay = basepay;
wagecal.OtPay = otpay;
wagecal.LeavePay = leavepay;
int result = wagecal.Calculate();
```

```
var basepay = CalculateBaseFactory.CalBasePay;
var leavepay = CalculateLeavePayFactory.CalLeavePay;
var otpay = CalculateOtFactory.CalOtPay;

var wagecal = new CalculateWage();
wagecal.BasePay = basepay;
wagecal.OtPay = otpay;
wagecal.LeavePay = leavepay;
int result = wagecal.Calculate();
```

Use Unity

- 自動掃描註冊

```
var container = new UnityContainer(); container.RegisterTypes(
AllClasses.FromLoadedAssemblies(),  //掃描目前已經載入的全部組件
WithMappings.FromAllInterfaces,  //找尋所有的介面
getName: WithName.TypeName);
```

Summary

- 透過介面設計去除類別間的相依性
- 提升可維護性
- DI 模式提供更好的系統彈性
- 重構是隨時進行的
- 撰寫測試確保程式碼變動的品質

Q&A

參考資源

- https://msdn.microsoft.com/en-us/library/dn170424.aspx
- https://msdn.microsoft.com/en-us/library/dn507453.aspx