淺談 ASP.NET Caching 技術與實踐

twMVC #26



- 阿磊 Cheng-Ju Wu
- C#, ASP.NET MVC, WebAPI

開始前

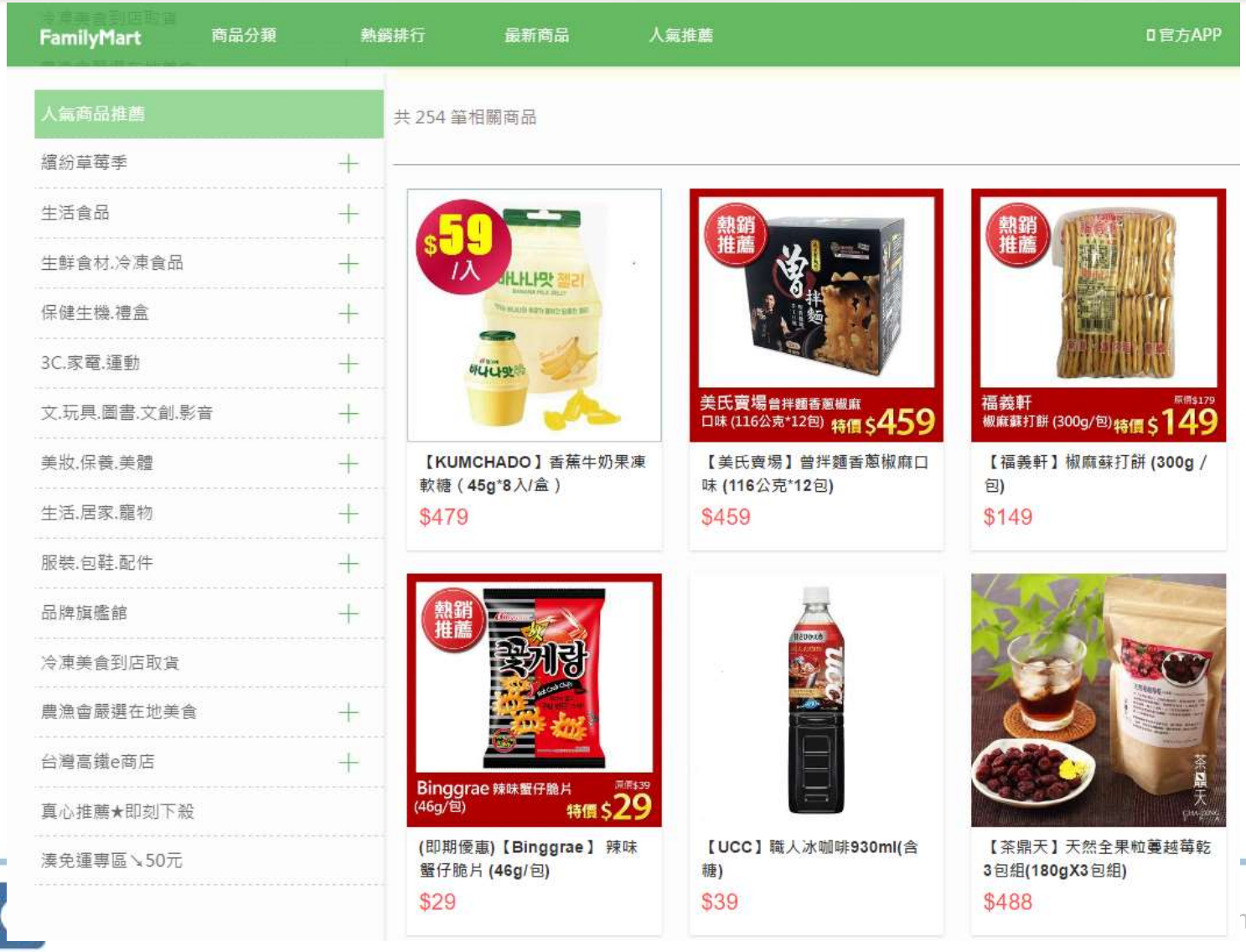
- 資料庫每一次的查詢都是不小的開銷，例如：
  - 連結的開啟
  - 執行查詢指令
  - 當資料庫與應用伺服器不在同一個伺服器上時，還必須有遠程調用、Socket的建立等開銷。

開始前

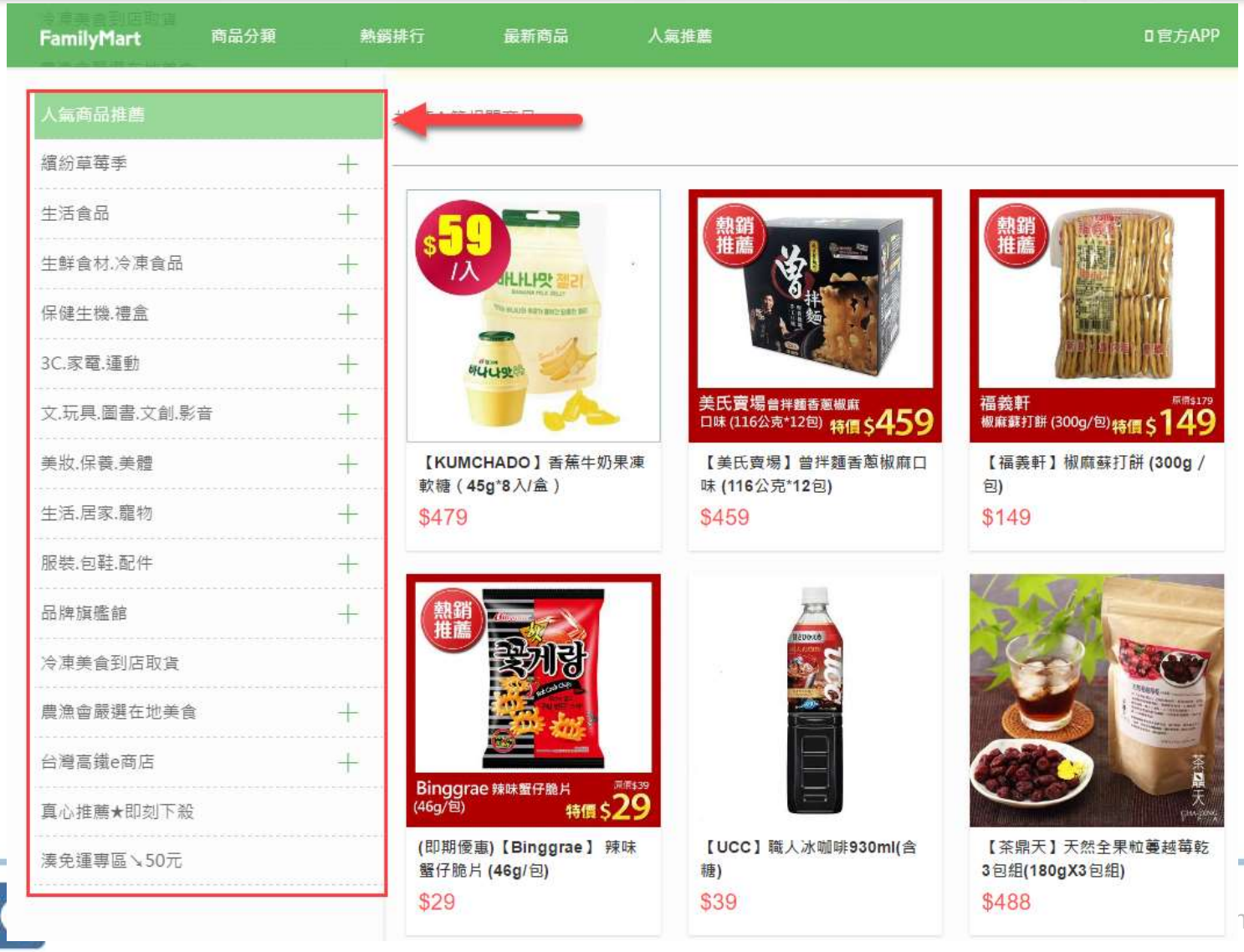
- 快取（Cache）是資料在記憶體中的臨時容器，從資料庫中讀取的資料在快取中會有一份臨時拷貝。

- 查詢某個資料時，先在快取中尋找是否有相對應的拷貝。
  - 有，直接返回資料。
  - 無，從資料庫查詢，返回資料。
  - 如此提昇應用程式讀取資料時的效能。

開始前


FamilyMart
商品分類
熱銷排行
最新商品
人氣推薦
官方APP
人氣商品推薦
繽紛草莓季
生活食品
生鮮食材.冷凍食品
保健生機.禮盒
3C.家電.運動
文.玩具.圖書.文創.影音
美妝.保養.美體
生活.居家.寵物
服裝.包鞋.配件
品牌旗艦館
冷凍美食到店取貨
農漁會嚴選在地美食
台灣高鐵e商店
真心推薦★即刻下殺
湊免運專區↘50元
共 254 筆相關商品
$59 /入
【KUMCHADO】香蕉牛奶果凍軟糖（45g*8入/盒）
$479
熱銷推薦
美氏賣場曾拌麵香蔥椒麻口味 (116公克*12包) 特價 $459
【美氏賣場】曾拌麵香蔥椒麻口味 (116公克*12包)
$459
熱銷推薦
福義軒 椒麻蘇打餅 (300g/包) 原價$179 特價 $149
【福義軒】椒麻蘇打餅 (300g / 包)
$149
熱銷推薦
Binggrae 辣味蟹仔脆片 (46g/包) 原價$39 特價 $29
(即期優惠)【Binggrae】 辣味蟹仔脆片 (46g/包)
$29
【UCC】職人冰咖啡930ml(含糖)
$39
【茶鼎天】天然全果粒蔓越莓乾3包組(180gX3包組)
$488

開始前


FamilyMart
商品分類
熱銷排行
最新商品
人氣推薦
官方APP
人氣商品推薦
繽紛草莓季
生活食品
生鮮食材.冷凍食品
保健生機.禮盒
3C.家電.運動
文.玩具.圖書.文創.影音
美妝.保養.美體
生活.居家.寵物
服裝.包鞋.配件
品牌旗艦館
冷凍美食到店取貨
農漁會嚴選在地美食
台灣高鐵e商店
真心推薦★即刻下殺
湊免運專區↘50元
$59 /入
【KUMCHADO】香蕉牛奶果凍軟糖（45g*8入/盒）
$479
熱銷推薦
美氏賣場曾拌麵香蔥椒麻口味 (116公克*12包) 特價 $459
【美氏賣場】曾拌麵香蔥椒麻口味 (116公克*12包)
$459
熱銷推薦
福義軒 椒麻蘇打餅 (300g/包) 原價$179 特價 $149
【福義軒】椒麻蘇打餅 (300g / 包)
$149
熱銷推薦
Binggrae 辣味蟹仔脆片 (46g/包) 原價$39 特價 $29
(即期優惠)【Binggrae】 辣味蟹仔脆片 (46g/包)
$29
【UCC】職人冰咖啡930ml(含糖)
$39
【茶鼎天】天然全果粒蔓越莓乾3包組(180gX3包組)
$488

開始前

- 快取使用情境
  - 商店分類選單，時常出現於整個網站。
  - 資料異動頻率不高。
  - 將資料存放到快取，就能快速回應資料。

開始前

- 快取使用情境
  - 商店分類選單，時常出現於整個網站。
  - 資料異動頻率不高。
  - 將資料存放到快取，就能快速回應資料。

- 那麼就開始主題吧

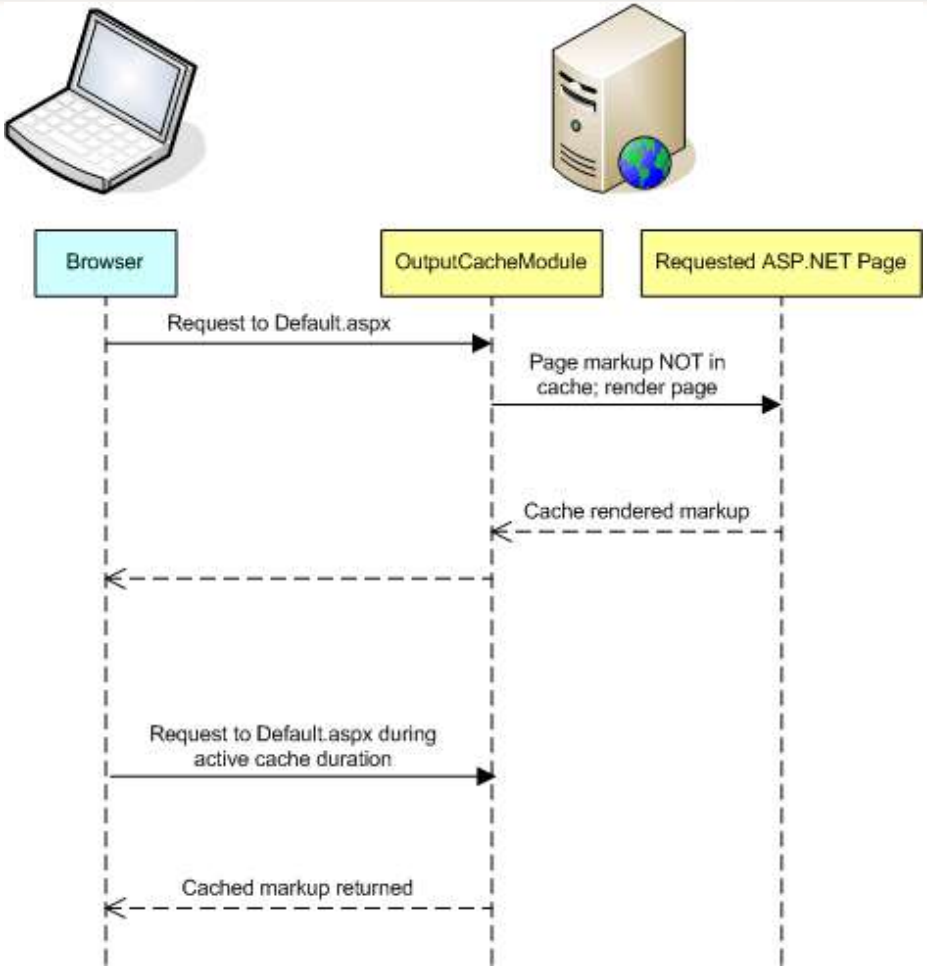
綱要

- 快取的類型，常見快取的作法
- 快取的設計面向
  - CacheProvider
  - DataCacheService
- ASP.NET Core 快取實踐

快取的類型

- Page Output Caching
  - OutputCacheAttribute
  - RedisOutputCacheProvider
- Application Caching

OutputCache



OutputCache

- `[OutputCache(Duration = 60)]`

```
[OutputCache(Duration = 60)]
public ActionResult Index()
{
   var employees = from e in db.Employees
                   orderby e.ID
                   select e;

   return View(employees);
}
```

- `[OutputCache(Location = OutputCacheLocation.ServerAndClient,`
  `Duration = 600, VaryByParam = "Id")]`

收工

- 設定完成後，這樣就有快取囉。
- 方便、快速，好簡單。



- 這麼冷的天氣，可以快點下班回家。

但...

- 實務上，不會這麼簡單...



OutputCache

- 實務上不會把設定分散於各個 Action。
- 造成管理、維護不便。
- 可透過 OutputCacheProfile 設定與維護。

```
<caching>
   <outputCacheSettings configSource="Cache.config" />
</caching>
```

OutputCache

- OutputCacheProfile

```
<outputCacheSettings>
  <outputCacheProfiles>
    <add name="JS" enabled="false" duration="1209600" varyByParam="isInput;v" />
    <add name="Category" enabled="false" duration="1200" varyByParam="id;v;r" />
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="adCode;datetime;type;shopId;r" va
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="shopId;datetime;r" />
    <add name="WebAPI.                    " enabled="true" duration="60" varyByCustom="isMobile;isOfficial" varyB
    <add name="WebAPI.                    " enabled="false" duration="30" varyByParam="id;v;r" varyByCustom="isOfficial" /
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="shopId;r" varyByCustom="isOffici
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="r;v" />
    <add name="WebAPI.                    " enabled="false" duration="30" varyByParam="cid;orderby;startIndex;maxCount" />
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="id;cid;h;o;r" />
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="id;r;source;shopId" />
    <add name="WebAPI.                    " enabled="true" duration="120" varyByParam="width;height;type;id;position;v" />
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="shopId" varyByCustom="isOfficial" />
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="id" />
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="shopId;OSType" varyByCustom="isOffi
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="cidList" />
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="platform;appVersion;osVersion;shopId
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="Id" />
    <add name="WebAPI.                    " enabled="false" duration="60" varyByCustom="isOfficial" varyByPar
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="shopId" varyByCustom="isOfficial
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="cidList" />
    <add name="WebAPI.                    " enabled="false" duration="60" varyByParam="shopId;cidList" varyBy
```

OutputCache

- 如此，線上服務的 Web 伺服器各自有快取資料



快取資料



快取資料



快取資料

OutputCache

- 有可能因 Web 伺服器被訪問的機率不同，造成資料不一致。



OutputCache

- 有可能因 Web 伺服器被訪問的機率不同，造成資料不一致。



- 那怎麼做，來達到資料一致性呢?

OutputCache 之 Redis

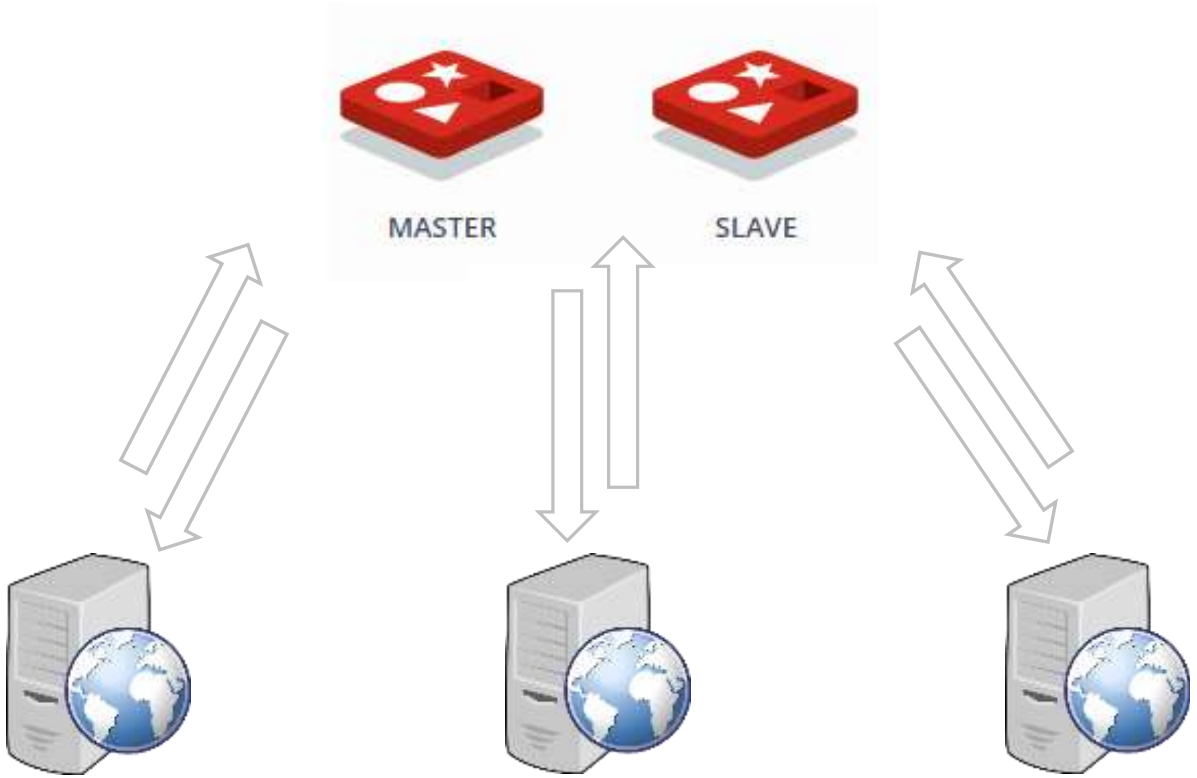
- RedisOutputCacheProvider
- 將快取資料從本地端轉移到 Redis Server

```
Install-Package Microsoft.Web.RedisOutputCacheProvider
```

```
<caching>
   <outputCacheSettings configSource="Cache.config" />
      <outputCache defaultProvider="RedisOutputCache">
         <providers>
            <clear />
            <add name="RedisOutputCache“
                 type="Microsoft.Web.Redis.RedisOutputCacheProvider"
                 connectionString=“xxxxx"
                 applicationName="OutputCache:Dev:WebAPI:" />
         </providers>
   </outputCache>
 </caching>
```

OutputCache 之 Redis

- 集中資料於 Redis Server (單台或分散式)





demo

CacheProvider

CacheProvider

- 發想於 nopCommerce https://goo.gl/khyY0e
- ICacheProvider
  - MemoryCacheProvider
  - NullCacheProvider
  - RedisCacheProvider
  - …

CacheProvider

- 發想於 nopCommerce https://goo.gl/khyY0e
- ICacheProvider
  - MemoryCacheProvider
  - NullCacheProvider
  - RedisCacheProvider
  - …

CacheProvider

- ICacheProvider 方法簽章

```
public interface ICacheProvider
{
        T Get<T>(string key);

        void Set(string key, object data, int cacheTime);

        bool IsSet(string key);

        void Remove(string key);

        void RemoveByPattern(string pattern);

        void Clear();
}
```

MemoryCacheProvider

- MemoryCacheProvider 實作

```
public partial class MemoryCacheProvider : ICacheProvider
{
        protected ObjectCache Cache { get { return MemoryCache.Default; } }

        public T Get<T>(string key) { return (T)Cache[key]; }

        public void Set(string key, object data, int cacheTime)
        {
            if (data == null) return;
            var policy = new CacheItemPolicy();
            policy.AbsoluteExpiration = DateTime.Now + TimeSpan.FromSeconds(cacheTime);
            Cache.Remove(key);
            Cache.Add(new CacheItem(key, data), policy);
        }
        .....
}
```

MemoryCacheProvider + Autofac

- MemoryCacheProvider + Autofac 設定

```
public class ProviderModule : Module
{
    protected override void Load(ContainerBuilder builder)
    {
        var cacheProviderType = ConfigurationManager.AppSettings.Get("CacheProviderType");

        builder
            .RegisterType(Type.GetType(cacheProviderType))
            .As<ICacheProvider>()
            .InstancePerLifetimeScope();
    }
}
```

MemoryCacheProvider + Autofac

- AppSettings 配置

```
<appSettings>
    <!-- CacheProvider  -->
    <add key="CacheProviderType" value="twmvc26samples.MemoryCacheProvider, twmvc26samples" />

    <!--<add key="CacheProviderType" value="twmvc26samples.NullCacheProvider, twmvc26samples" />-->
    <!--<add key="CacheProviderType" value="twmvc26samples.RedisCacheProvider, twmvc26samples" />-->
</appSettings>
```

CacheProvider 使用

```
public ActionResult Index()
{
    var cacheTime = 3;
    List<Employee> result = null;
    //// 1.從快取層取得資料
    var cacheData = this._cacheProvider.Get<List<Employee>>("Key");
    //// 2.找不到快取資料
    if (cacheData == null)
    {
        //// 3.從服務層取得資料
        result = this._employeeService.GetAll();
        //// 4.將資料設定到快取
        this._cacheProvider.Set("Key", result, cacheTime);
    }
    else
    {
        //// 3.快取有資料，直接回傳
        result = cacheData;
    }
    return View(result);
}
```

CacheProvider 使用

```
public ActionResult Index()
{
    var cacheTime = 3;
    List<Employee> result = null;
    //// 1.從快取層取得資料
    var cacheData = this._cacheProvider.Get<List<Employee>>("Key");
    //// 2.找不到快取資料
    if (cacheData == null)
    {
        //// 3.從服務層取得資料
        result = this._employeeService.GetAll();
        //// 4.將資料設定到快取
        this._cacheProvider.Set("Key", result, cacheTime);
    }
    else
    {
        //// 3.快取有資料，直接回傳
        result = cacheData;
    }
    return View(result);
}
```

取資料

Demo

CacheProvider

- 當應用程式套用快取模組，資料流與快取強耦合。
- 如此「服務層」無法在「沒有快取模組」下單獨運作。
- 本地端，開發/偵錯情境
  - 關注點分離?
  - 在正式邏輯中，想要沒有快取時，只能先暫時拔掉快取模組。
  - 是否發生過開發/偵錯完成後，忘了補回快取模組?

CacheProvider 使用

```
public ActionResult Index()
{
    var cacheTime = 3;
    List<Employee> result = null;
    //// 1.從快取層取得資料
    var cacheData = this._cacheProvider.Get<List<Employee>>("Key");
    //// 2.找不到快取資料
    if (cacheData == null)
    {
        //// 3.從服務層取得資料
        result = this._employeeService.GetAll();
        //// 4.將資料設定到快取
        this._cacheProvider.Set("Key", result, cacheTime);
    }
    else
    {
        //// 3.快取有資料，直接回傳
        result = cacheData;
    }
    return View(result);
}
```

要測試就要拔掉 Cache Code ？

要測試就要拔掉 Cache Code ？
翻桌會比較快...



Graph from https://goo.gl/1gdS5S

那就來看看有什麼解決方法

NullCacheProvider

- NullCacheProvider 實作

```
public partial class NullCacheProvider : ICacheProvider
{
        public T Get<T>(string key)  {  return default(T);  }

        public void Set(string key, object data, int cacheTime)  {  }

        public bool IsSet(string key)  {  return false;  }

        public void Remove(string key)  {  }

        public void RemoveByPattern(string pattern)  {  }

        public void Clear()  {  }
}
```

NullCacheProvider + Autofac

- 切換註冊Type

```
<appSettings>
    <!-- CacheProvider  -->
    <!--<add key="CacheProviderType" value="twmvc26samples.MemoryCacheProvider, twmvc26samples" />-->

    <add key="CacheProviderType" value="twmvc26samples.NullCacheProvider, twmvc26samples" />
    <!--<add key="CacheProviderType" value="twmvc26samples.RedisCacheProvider, twmvc26samples" />-->
</appSettings>
```

- 注入NullCache，如此不用手動拔除邏輯中的快取模組。

Demo

NullCacheProvider 價值

- 透過切換，讓功能回到本質的服務。進而偵錯與開發。
- 透過切換，不需要伸手弄髒既有的程式邏輯。

NullCacheProvider 價值

- 透過切換，讓功能回到本質的服務。進而偵錯與開發。
- 透過切換，不需要伸手弄髒既有的程式邏輯。

- 此時，全站的 `CacheProvider` 都以 `NullCache` 實作。
- 似乎，還可以更靈活!



demo

DataCacheService

DataCacheService

- IDataCacheService 方法簽章

```
public interface IDataCacheService
{
    /// <summary>
    /// 取得Redis快取內容
    /// </summary>
    T GetRedisCacheData<T>(string cacheName, string cacheType, string key, Func<T> source, int expirationSecond = 1800,
bool enabledCache = true, bool cleanCache = false) where T : class;

    /// <summary>
    /// 取得Memory快取內容
    /// </summary>
    T GetMemoryCacheData<T>(string cacheName, string cacheType, string key, Func<T> source, int
memoryCacheExpirationSecond = 10, bool enabledCache = true, bool cleanCache = false) where T : class;
}
```

DataCacheService

```
public class DataCacheService : IDataCacheService
{
    public T GetMemoryCacheData<T>(string cacheName, string cacheType, string key, Func<T> source, int
memoryCacheExpirationSecond = 10, bool enabledCache = true, bool cleanCache = false) where T : class
    {
        T returnResult;
        //// 沒有開啟cache的話就直接回傳
        if (enabledCache == false)
        {
            returnResult = source();
            return returnResult;
        }

        string fullCacheName = string.Format("{0}-{1}-{2}", cacheName, cacheType, key);
        var memoryCache = MemoryCache.Default;

        if (cleanCache)  {  memoryCache.Remove(fullCacheName);  }

        var memoryCacheObject = memoryCache.Get(fullCacheName);

        //// 找不到Cache時直接取得資料
        if (memoryCacheObject == null || cleanCache)  {  ....  }
        ....
        return returnResult;
    }
}
```

DataCacheService

```
public class DataCacheService : IDataCacheService
{
    public T GetMemoryCacheData<T>(string cacheName, string cacheType, string key, Func<T> source, int
memoryCacheExpirationSecond = 10, bool enabledCache = true, bool cleanCache = false) where T : class
    {
        T returnResult;
        //// 沒有開啟cache的話就直接回傳
        if (enabledCache == false)
        {
            returnResult = source();
            return returnResult;
        }

        string fullCacheName = string.Format("{0}-{1}-{2}", cacheName, cacheType, key);
        var memoryCache = MemoryCache.Default;

        if (cleanCache) { memoryCache.Remove(fullCacheName); }

        var memoryCacheObject = memoryCache.Get(fullCacheName);

        //// 找不到Cache時直接取得資料
        if (memoryCacheObject == null || cleanCache) { …. }
        ….
        return returnResult;
    }
}
```

DataCacheService

- 使用方式

```
public ActionResult Index()
{
    var employeeList = this._dataCacheService.GetMemoryCacheData<List<Employee>>(
        "ServiceName",
        "TypeName",
        "Key",
        () =>
        {
            var result = this._employeeService.GetAll();
            return result;
        },
        expirationSecond,
        enabledCache,
        cleanCache);

    return View(employeeList);
}
```

DataCacheService

- 使用方式

```
public ActionResult Index()
{
    var employeeList = this._dataCacheService.GetMemoryCacheData<List<Employee>>(
        "ServiceName",
        "TypeName",
        "Key",
        () =>
        {
            var result = this._employeeService.GetAll();
            return result;
        },
        expirationSecond,
        enabledCache,
        cleanCache);

    return View(employeeList);
}
```

取資料

開關

Demo



demo

ASP.NET Core Caching

ASP.NET Core Caching

- GitHub

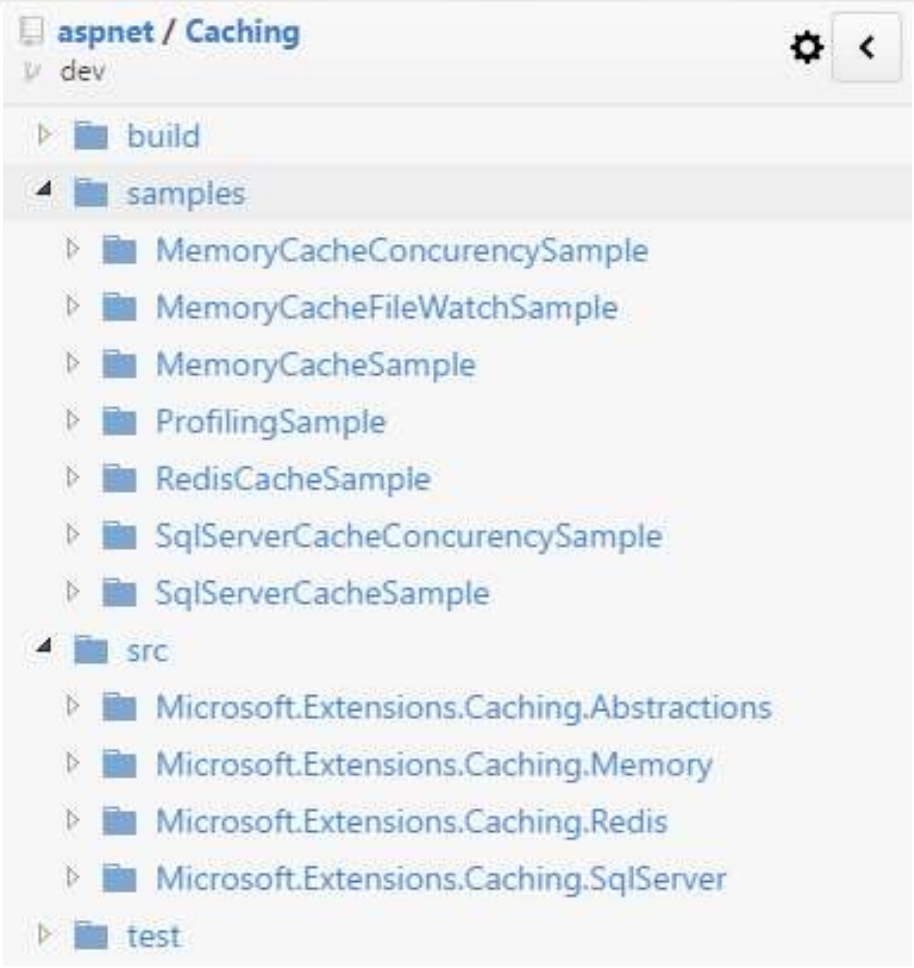
  https://github.com/aspnet/Caching

- Implementations

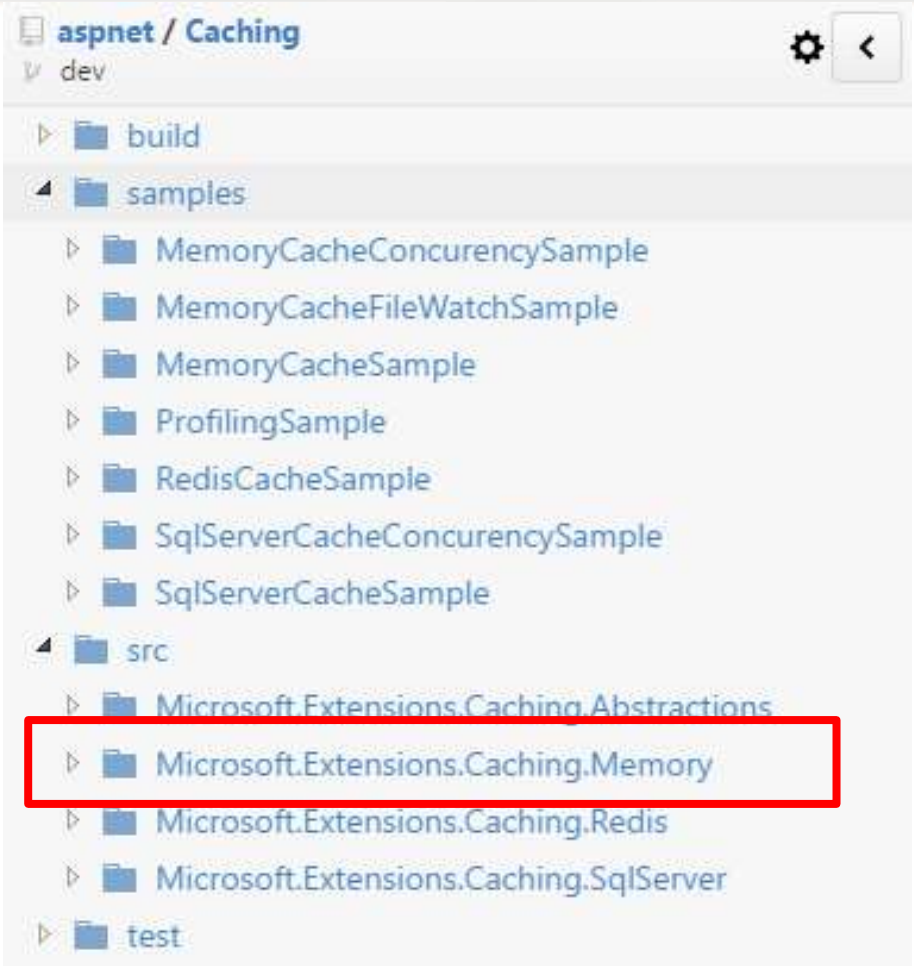
  In-Memory, Microsoft SQL Server, and Redis.

- 以前微軟沒提供，大家各有各做法，現在有了標準實作。

ASP.NET Core Caching



ASP.NET Core Caching



How to using IMemoryCache

- NuGet

```
Install-Package Microsoft.Extensions.Caching.Memory
```

How to using IMemoryCache

- Call AddMemoryCache in ConfigureServices

```
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.DependencyInjection;

public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMemoryCache();
        services.AddMvc();
    }
}
```

How to using IMemoryCache

- IMemoryCace instance in the constructor

```
public class HomeController : Controller
{
    private IMemoryCache _cache;

    public HomeController(IMemoryCache memoryCache)
    {
        _cache = memoryCache;
    }
}
```

ASP.NET Core Caching

- 接下來就以「取得員工資料」的範例來說明。



demo

總結 – 思考

總結 – 思考

- 快取是非常有用的工具，但也容易被濫用。
- 調校邏輯所花費的時間，其實跟加入快取模組差不多。
- 出現效能瓶頸時，優先考慮快取，會導致效能問題被掩蓋，無法真正解決問題。
- 不到最後一刻不要使用快取，而是優先考序使用其他方式優化應用程式效能。

Blog 是記錄知識的最佳平台



感謝 Jetbrains 贊助贈品



ReSharper

https://www.jetbrains.com/resharper/

感謝 OzCode 贊助贈品



http://www.oz-code.com/

業界師資、實戰教學



http://skilltree.my

業界師資、實戰教學



https://mvc.tw

謝謝各位

https://mvc.tw

- 本投影片所包含的商標與文字皆屬原著作者所有。
- 本投影片使用的圖片皆從網路搜尋。
- 本著作係採用姓名標示-非商業性-相同方式分享 3.0 台灣授權。閱讀本授權條款，請到 http://creativecommons.org/licenses/by-nc-sa/3.0/tw/，或寫信至Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.