



## 範例教學：使用 ASP.NET MVC 打造 WebAPI 服務

📅 2018-10-06 09:42 PM 💬 18 👁 56,459

### 前言

[閒聊 - Web API 是否一定要 RESTful?](#) 一文提到我個人偏好用 ASP.NET MVC 寫 WebAPI，讀者 Mark 留言希望能有簡單範例參考。文章將示範用 ASP.NET MVC 從無到有打造一個簡單 WebAPI 服務，提供給初學 ASP.NET MVC 不知如何下手的新手參考。註：展示概念，省略了一些實務應有環節以免複雜失焦，如要應用於正式對外商轉，需將必要的安全與管理功能補齊。

希望文章裡的說明夠詳細，足以讓新同學們依步驟徒手寫出自己的 WebApi，但為防有人不幸卡關，範例專案我已放上 [Github](https://github.com/darkthread/WebApiWithAspNetMvcDemo) <https://github.com/darkthread/WebApiWithAspNetMvcDemo> 當作急救包，希望用不到。XD

### 本文開始

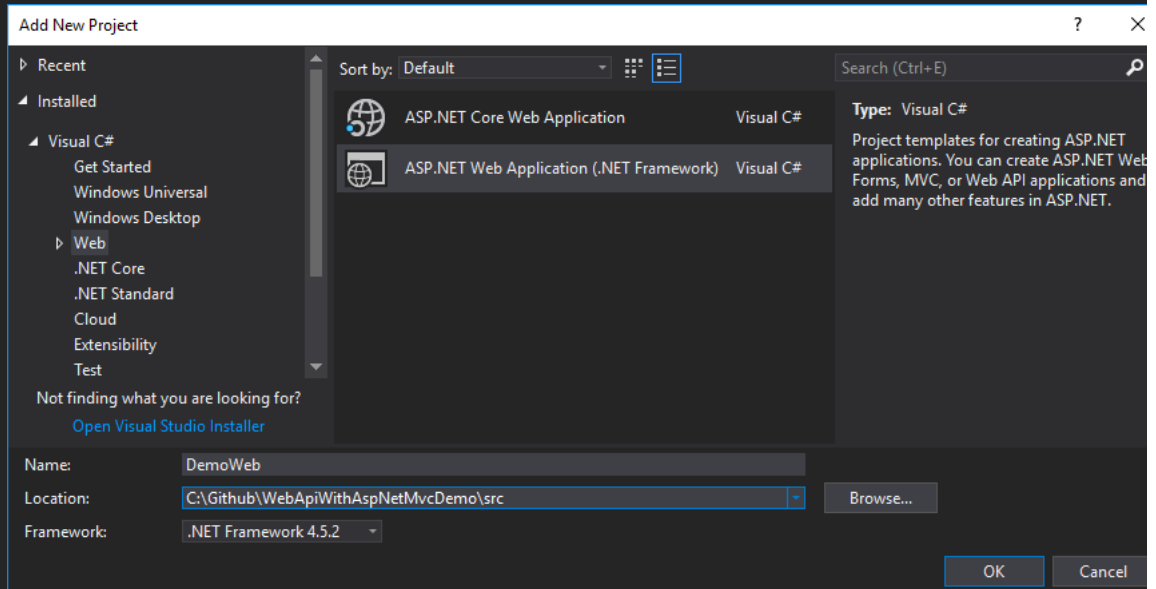
來到正題。假設我有個加密演算法想寫成 WebAPI，介面如下：

1. EncryptString，傳入加密金鑰與待加密文字：encKey (字串), rawText (字串)，API 產生加密資料以 byte[] 回傳
2. BatchDecryptString，傳入解密金鑰 encKey 與多個加密資料 (List<byte[]>) API 批次解密，結果以 List 傳回

第一個 WebAPI 方法用一般 POST Form 傳參數即可，第二個 WebAPI 傳送的參數較複雜，我選擇另外定義參數物件，JSON 從 POST 內文送出，這也是實務常見做法。故意安排不同形式，藉此展示兩種不同呼叫方式的實作。

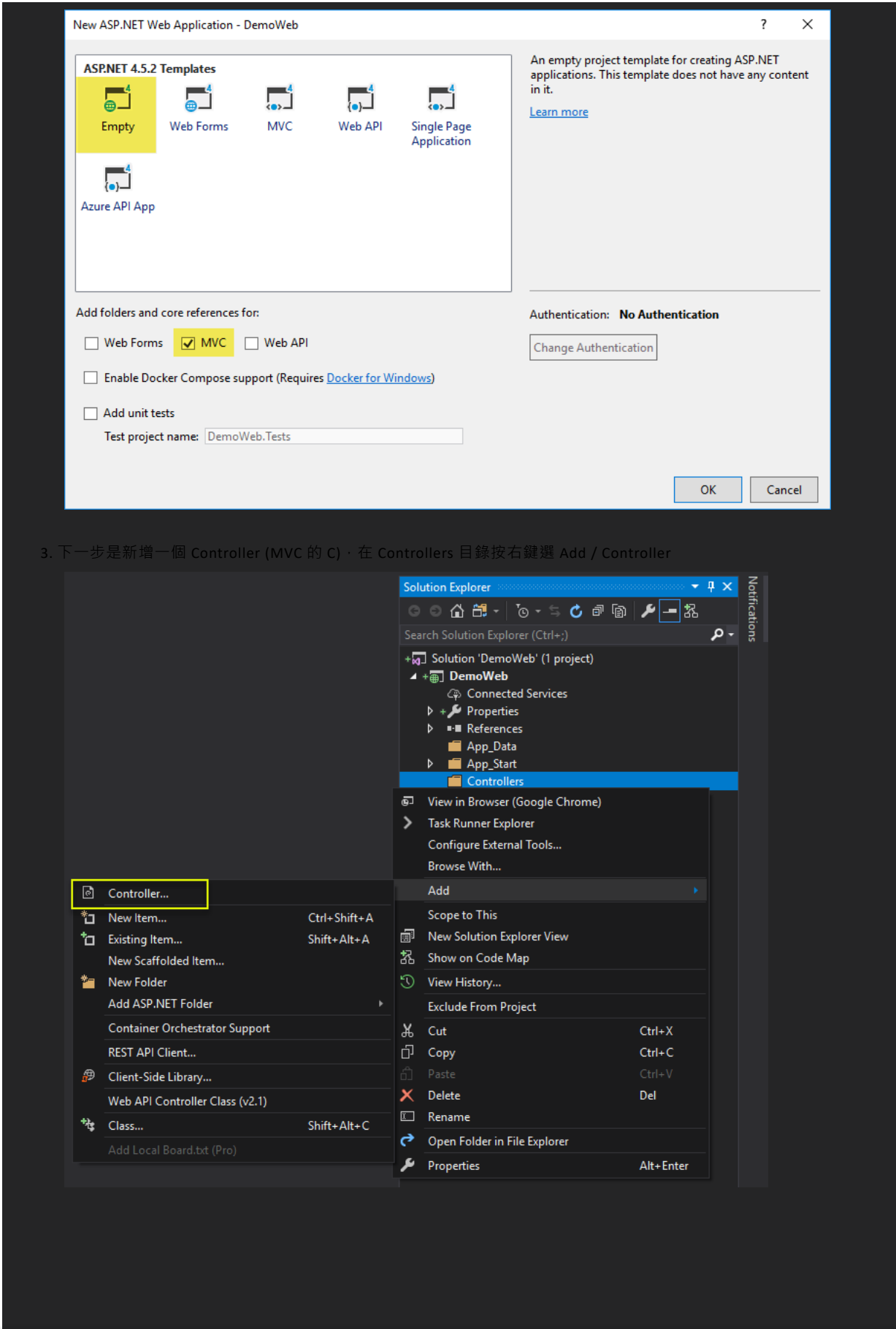
以下示範如何用 ASP.NET MVC 建立上述 WebAPI：

1. 使用 VS2015/VS2017 建立一個 ASP.NET Web Application (別問我如果用 VS2013/VS2012/VS2010 該怎麼辦，我不想逆天)



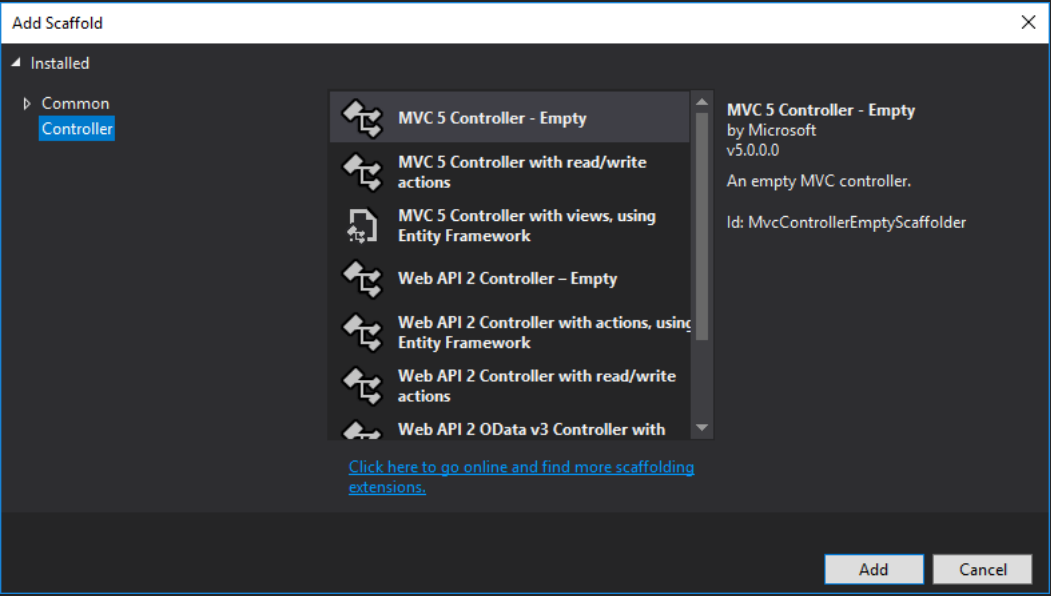
2. ASP.NET 5 起採用 One ASP.NET 概念，建立專案選 Empty 模版省去移除多餘項目的功夫，啟用項目則勾選 MVC 就好：(我不愛用 ASP.NET MVC 內建 WebAPI 功能的原因可參考：[閒聊 - Web API 是否一定要 RESTful?](#))



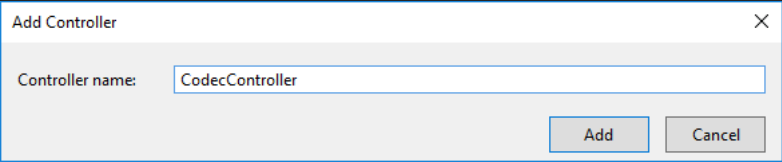


3. 下一步是新增一個 Controller (MVC 的 C) · 在 Controllers 目錄按右鍵選 Add / Controller

4. 新增 Controller 時選 MVC 5 Controller – Empty 即可。



Controller 名字很重要，如果希望 WebAPI 的 URL 是 http://myserver/CodecApi/EncryptString，Controller 就要取名為 CodecApiController



建好的 Controller 會預建一個空白 Index() 方法以及 Views/CodecApi/Index.cshtml，我們都用不到，請直接刪掉。

5. 加解密部分非示範重點就不多解釋，我做了一個 Models/CodecModule.cs 實做加解密。(商業邏輯類別統一放在 Models



```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Web;

namespace DemoWeb.Models
{
    public class CodecModule
    {
        public static byte[] EncryptString(string encKey, string rawText)
        {
            return DESEncrypt(encKey, Encoding.UTF8.GetBytes(rawText));
        }

        public static List<string> DecryptData(string encKey, List<byte[]> data)
        {
            return data.Select(o =>
            {
                return Encoding.UTF8.GetString(DESDecrypt(o, encKey));
            }).ToList();
        }

        //REF: https://dotblogs.com.tw/supershowwei/2016/01/11/135230
        static byte[] HashByMD5(string source)
        {
            MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();

            return md5.ComputeHash(Encoding.UTF8.GetBytes(source));
        }

        static byte[] DESEncrypt(string key, byte[] data)
        {
            var des = new DESCryptoServiceProvider();
            Rfc2898DeriveBytes rfc2898 = new Rfc2898DeriveBytes(key, HashByMD5(key));
            des.Key = rfc2898.GetBytes(des.KeySize / 8);
            des.IV = rfc2898.GetBytes(des.BlockSize / 8);
            using (MemoryStream ms = new MemoryStream())
            using (CryptoStream cs = new CryptoStream(ms, des.CreateEncryptor(), CryptoStreamMode.Write))
            {
                cs.Write(data, 0, data.Length);
                cs.FlushFinalBlock();

                return ms.ToArray();
            }
        }

        static byte[] DESDecrypt(byte[] encData, string encKey)
        {
            DESCryptoServiceProvider des = new DESCryptoServiceProvider();
            Rfc2898DeriveBytes rfc2898 = new Rfc2898DeriveBytes(encKey, HashByMD5(encKey));
            des.Key = rfc2898.GetBytes(des.KeySize / 8);
            des.IV = rfc2898.GetBytes(des.BlockSize / 8);
            using (MemoryStream ms = new MemoryStream())
            using (CryptoStream cs = new CryptoStream(ms, des.CreateDecryptor(), CryptoStreamMode.Write))
            {
                cs.Write(encData, 0, encData.Length);
                cs.FlushFinalBlock();

                return ms.ToArray();
            }
        }
    }
}
```

6. WebAPI 通常會開匿名存取，安全管控要自己來，最基本的防禦是鎖定呼叫來源 IP，更嚴謹一點可要求呼叫端附上專屬 Key/Secret，更機車一點還可以限定某支 API Key 只能用於哪些 IP。



```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Web;

namespace DemoWeb.Models
{
    public class SecurityManager
    {
        //此處使用web.config設定允許存取來源IP，實務上可改用DB存放並加上管理介面
        private static string[] allowedClientIps =
            (ConfigurationManager.AppSettings["api:AllowedClientIps"] ?? string.Empty)
            .Split(',', ';');
        public static void Authorize(HttpRequestBase request)
        {
            if (!allowedClientIps.Contains(request.UserHostAddress))
                throw new ApplicationException("Client IP Denied");
            //如要求更嚴謹管控時可發放API Key，並要求附於Request Header
            //在此可檢查API Key是否合法，甚至API Key再綁定特定IP使用
            //...request.Cookies["X-API-Key"]...
        }
    }
}
```

配合 IP 管控需在 web.config 加入可呼叫的來源 IP，例如測試期限定本機可以這樣寫：

```
<add key="api:AllowedClientIps" value="::1;127.0.0.1"/>
```

7. WebAPI 傳回結果的標準格式為 JSON，ASP.NET MVC Controller 雖然內建 JSON 序列化函式 `Json()`，但用的是微軟自家的 `JavaScriptSerializer`，而非業界主流 - `Json.NET`，建議費點手腳換掉，這點是用 ASP.NET MVC 寫 WebAPI 少數較不便的地點 (ASP.NET MVC WebAPI 預設用 `Json.NET`，勝出) 置換 `Controller.Json()` 的細節可參考舊文：[ASP.NET MVC小改裝 - 以Json.NET JavaScriptSerializer](#) 先用 NuGet 安裝 `Json.NET` 再將文中的範例程式新增為 `Models/JsonNetController.cs`，`CodecApiController` 繼承 `Controller` 改為繼承 `JsonNetController` 後，之後呼叫 `Json()` 便會改使用 `Json.NET` 執行序列化。



```

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.Mvc;

namespace DemoWeb.Models
{
    public class JsonNetController : Controller
    {
        protected override JsonResult Json(object data, string contentType,
            Encoding contentEncoding, JsonRequestBehavior behavior)
        {
            if (behavior == JsonRequestBehavior.DenyGet
                && string.Equals(this.Request.HttpMethod, "GET",
                    StringComparison.OrdinalIgnoreCase))
            {
                //Call JsonResult to throw the same exception as JsonResult
                return new JsonResult();
            }
            return new JsonNetResult()
            {
                Data = data,
                ContentType = contentType,
                ContentEncoding = contentEncoding
            };
        }
    }

    public class JsonNetResult : JsonResult
    {
        public JsonSerializerSettings SerializerSettings { get; set; }
        public Formatting Formatting { get; set; }
        public JsonNetResult()
        {
            SerializerSettings = new JsonSerializerSettings();
        }
        public override void ExecuteResult(ControllerContext context)
        {
            if (context == null)
            {
                throw new ArgumentNullException("context");
            }
            HttpResponseBase response = context.HttpContext.Response;
            response.ContentType =
                !string.IsNullOrEmpty(ContentType) ? ContentType : "application/json";
            if (ContentEncoding != null)
            {
                response.ContentEncoding = ContentEncoding;
            }
            if (Data != null)
            {
                JsonTextWriter writer = new JsonTextWriter(response.Output)
                {
                    Formatting = Formatting
                };
                JsonSerializer serializer = JsonSerializer.Create(SerializerSettings);
                serializer.Serialize(writer, Data); writer.Flush();
            }
        }
    }
}

```

8. 另外，先前講過我習慣用統一的 JsonResult 物件回傳 WebAPI 結果，故要在 Models/ApiResult.cs 定義 ApiResult 物件，這泛型強化版(ApiResult)，可指定 Data 型別加上強型別保護。



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace DemoWeb.Models
{
    /// <summary>
    /// API呼叫時·傳回的統一物件
    /// </summary>
    public class ApiResult<T>
    {
        /// <summary>
        /// 執行成功與否
        /// </summary>
        public bool Succ { get; set; }
        /// <summary>
        /// 結果代碼(0000=成功·其餘為錯誤代號)
        /// </summary>
        public string Code { get; set; }
        /// <summary>
        /// 錯誤訊息
        /// </summary>
        public string Message { get; set; }
        /// <summary>
        /// 資料時間
        /// </summary>
        public DateTime DateTime { get; set; }
        /// <summary>
        /// 資料本體
        /// </summary>
        public T Data { get; set; }

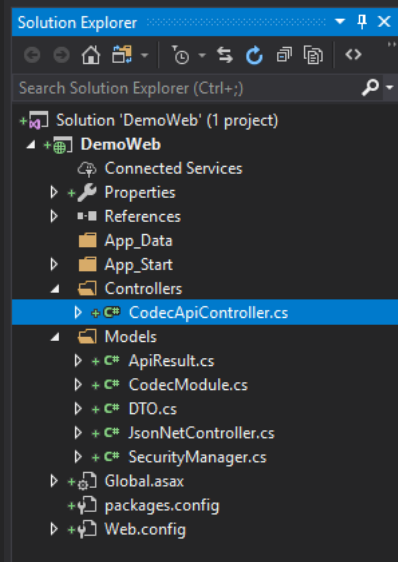
        public ApiResult() { }

        /// <summary>
        /// 建立成功結果
        /// </summary>
        /// <param name="data"></param>
        public ApiResult(T data)
        {
            Code = "0000";
            Succ = true;
            DateTime = DateTime.Now;
            Data = data;
        }
    }

    public class ApiError : ApiResult<object>
    {
        /// <summary>
        /// 建立失敗結果
        /// </summary>
        /// <param name="code"></param>
        /// <param name="message"></param>
        public ApiError(string code, string message)
        {
            Code = code;
            Succ = false;
            this.DateTime = DateTime.Now;
            Message = message;
        }
    }
}
```



9. 做到這裡，專案的主要檔案架構就差不多了：



10. 接著來寫Controller

```
using DemoWeb.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace DemoWeb.Controllers
{
    public class CodecApiController : JsonNetController
    {
        [HttpPost]
        public ActionResult EncryptString(string encKey, string rawText)
        {
            SecurityManager.Authorize(Request);
            return Json(new ApiResult<byte[]>{
                CodecModule.EncryptString(encKey, rawText)});
        }

        public class DecryptParameter
        {
            public string EncKey { get; set; }
            public List<byte[]> EncData { get; set; }
        }

        [HttpPost]
        public ActionResult BatchDecryptData(DecryptParameter decData)
        {
            SecurityManager.Authorize(Request);
            return Json(new ApiResult<List<string>>{
                CodecModule.DecryptData(decData.EncKey, decData.EncData)});
        }
    }
}
```

每個 ActionResult 方法對應一個 WebAPI 方法，加解密前要呼叫 SecurityManager.Authorize() 檢查客戶端 IP 確認是否為端。接著執行主要邏輯進行加解密，最後用 this.Json() 傳回 ApiResult。

加註 [HttpPost] 限定 POST 呼叫是好習慣。參考：[隱含殺機的GET式AJAX資料更新](#)

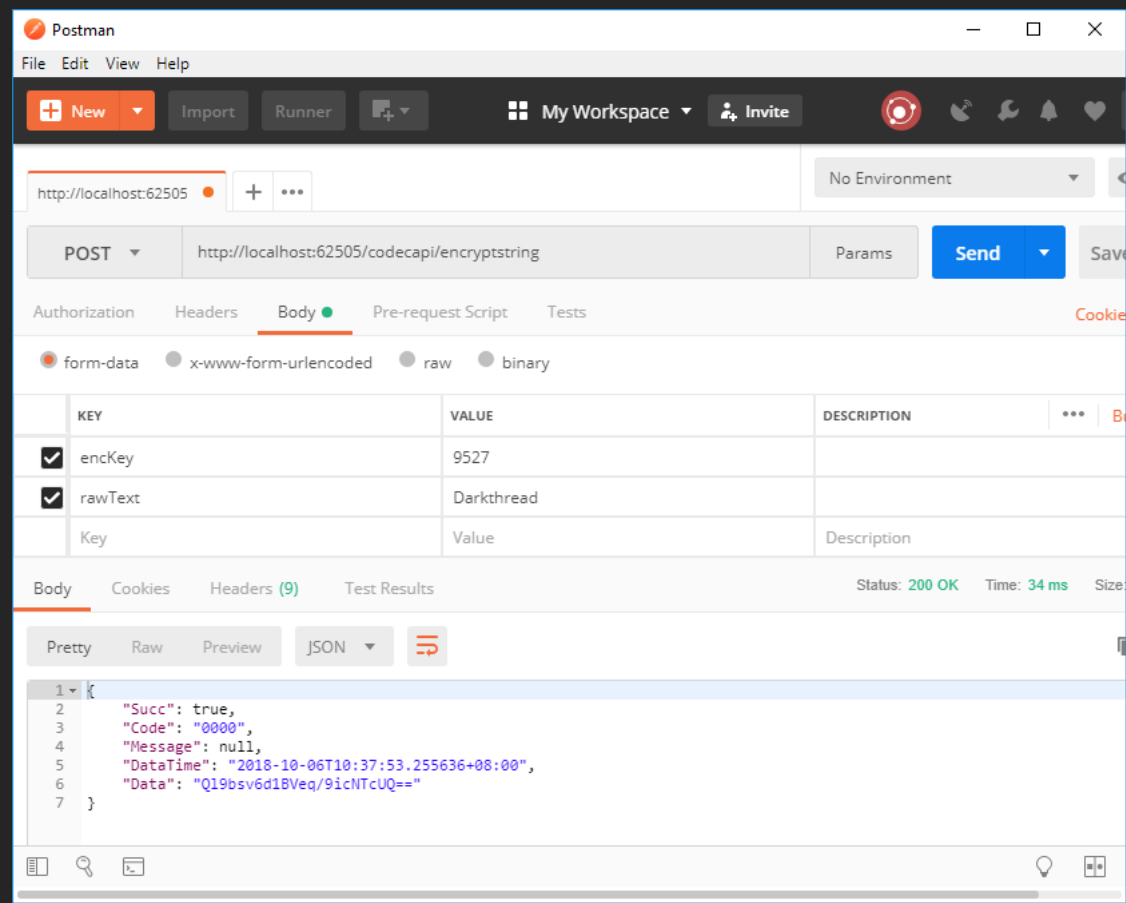
EncryptString() 宣告了 encKey, rawText 兩個字串參數，等下會示範如何傳進來。

BatchDecryptData() 的輸入參數我另外定義了 DecryptParamter 類別，這類複雜參數通常會以 JSON 格式作為 POST 本體節作法後面也會示範。





後可看到結果，成功!

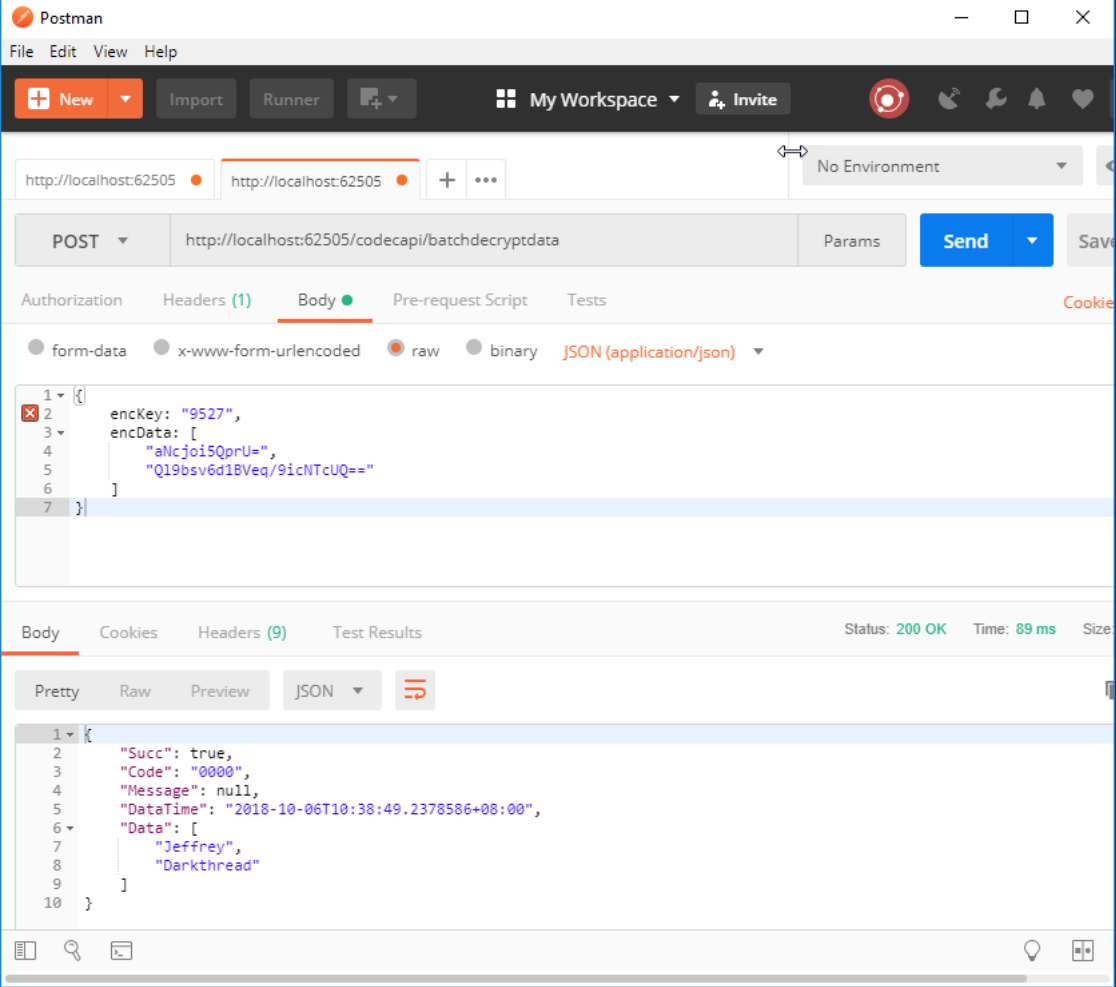


12. 測試 BatchDecryptData 要先準備一段 JSON 當作 Body :

```
{
  encKey: "9527",
  encData: [
    "aNcjoi5QprU=",
    "Q19bsv6d1BVeQ/9icNTcUQ=="
  ]
}
```



Body 型別選 raw，型別取 JSON(application/json)，按下 Send 也測試成功。



The screenshot shows the Postman interface with a POST request to `http://localhost:62505/codeapi/batchdecryptdata`. The request body is a JSON object:

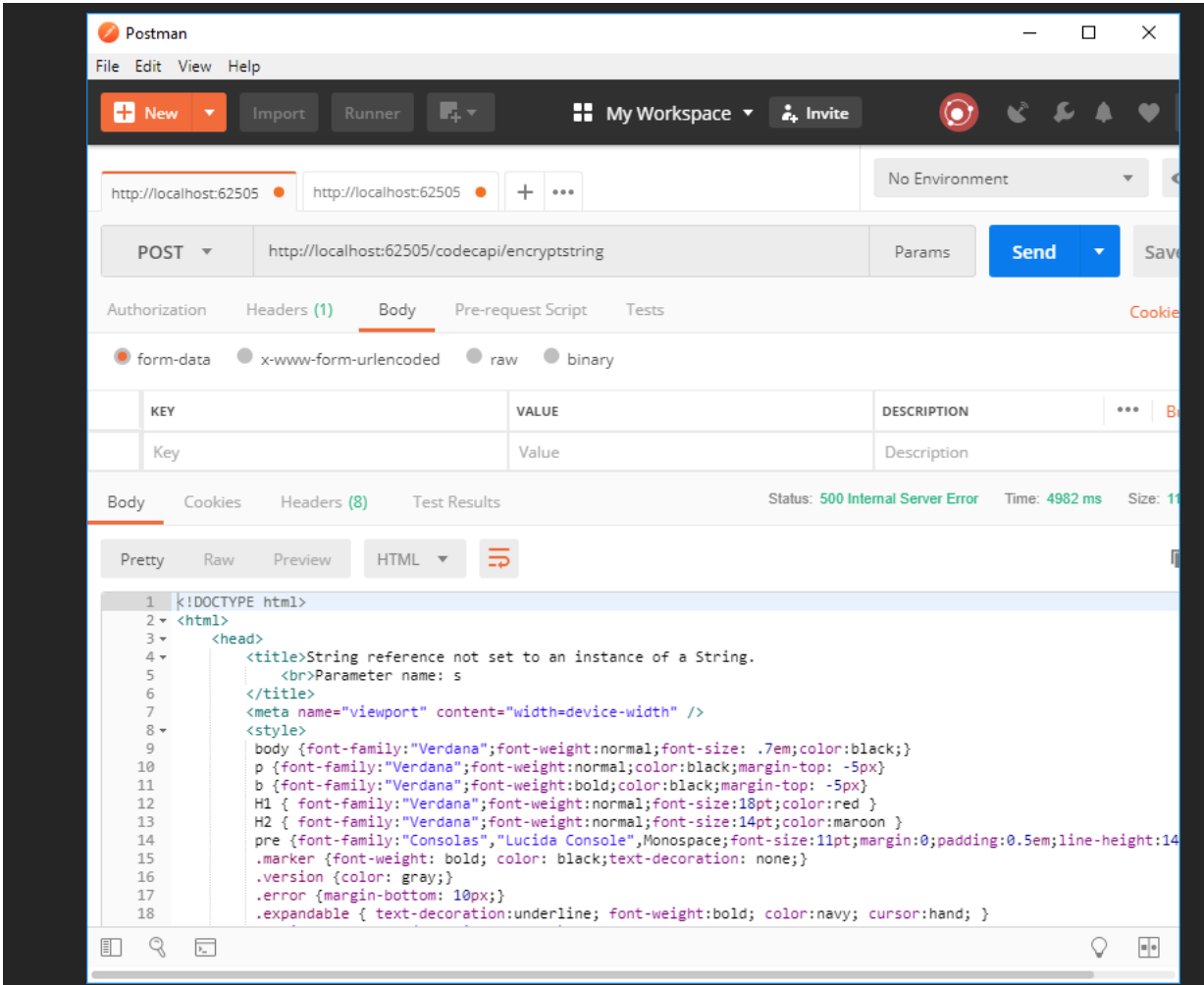
```
{  "encKey": "9527",  "encData": [    "aNcjoi5QprU=",    "Q19bsv6d18Veq/9icNTcUQ=="  ]}
```

The response status is 200 OK, and the response body is a JSON object:

```
{  "Succ": true,  "Code": "0000",  "Message": null,  "DateTime": "2018-10-06T10:38:49.2378586+08:00",  "Data": [    "Jeffrey",    "Darkthread"  ]}
```

13. 這樣我們就完成了基本雛型，以下再補充一些眉角。如果出錯了怎麼辦？例如故意不給 `encKey`，MVC 會噴出 HTTP 500，只知出錯很難從傳回結果抓出錯誤原因。

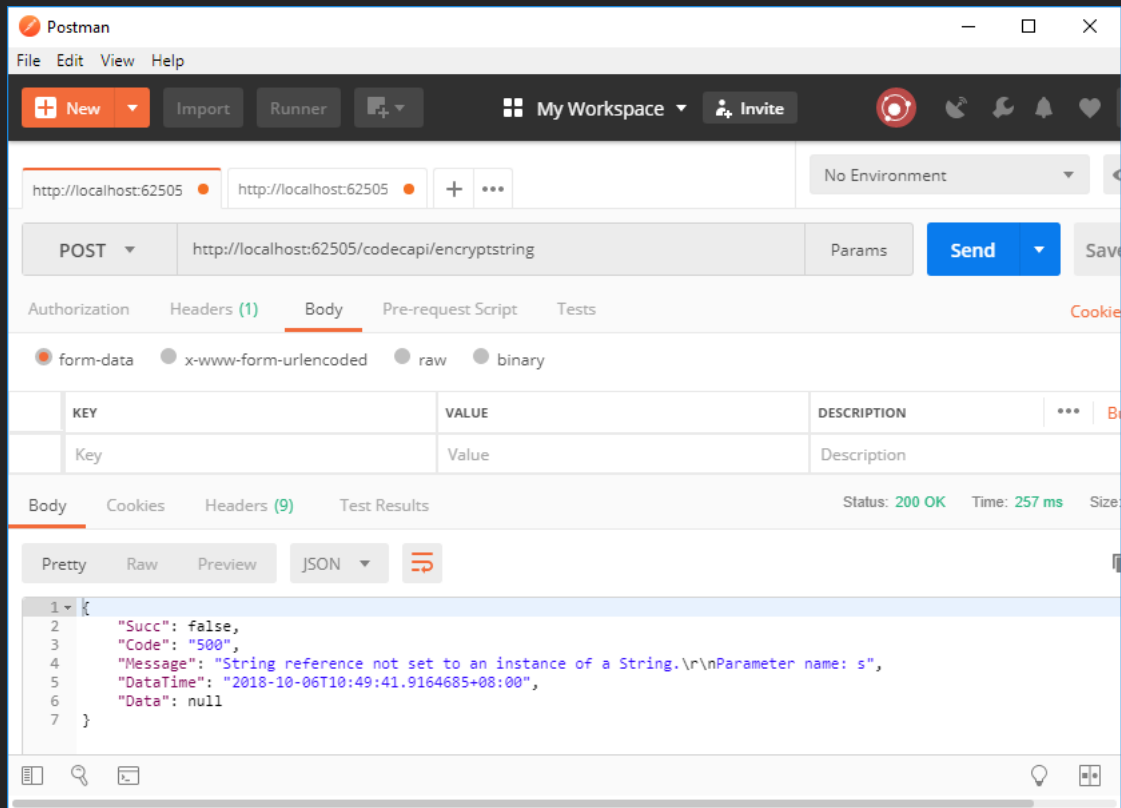




最簡單的做法是用 `try { ... } catch { ... }` 包住程式，`catch` 錯誤傳回 `ApiError`，像這個樣子：

```
[HttpPost]
public ActionResult EncryptString(string encKey, string rawText)
{
    try
    {
        SecurityManager.Authorize(Request);
        return Json(new ApiResult<byte[]>{
            CodecModule.EncryptString(encKey, rawText)});
    }
    catch (Exception ex)
    {
        return Json(new ApiError("500", ex.Message));
    }
}
```

如此，出錯時呼叫端一樣會拿到 `ApiResult`，並可由 `Code` 及 `Message` 取得錯誤資訊。



至此，這個雛型已符合規格具備最基本的 WebAPI 功能，但要應用在實務上還有不少改進空間，以下是一些努力方向：

1. 每個 Action 都加 `SecurityManager.Authorize(Request)` 跟 `try ... catch ...` 太醜，可考量用 `ActionFilter` 讓程式優雅一點~  
參考：[ASP.NET MVC Filter練習-限定本機存取](#)、[ASP.NET MVC实现IExceptionFilter接口编写自定义异常处理过滤器](#)  
<<https://shiyousan.com/post/635833789557065314>>
2. 出錯時回傳 `Exception.Message` 讓呼叫端知道原因，但想抓出問題需要更多 `Exception` 的細節，因此在 `try ... catch` 拋回要設法將完整的錯誤資訊(包含 `InnerException`)及環境細節(`Request.Url`、`User-Agent`、`IP...` 等)保存下來，參考：  
» [try... catch 造成錯誤資訊遺失的案例](#)  
» [實戰小技巧 - .NET Exception Message、InnerException 與 ToString\(\)](#)
3. 關於 WebAPI 存取權限控管，除了自幹也可引用業界標準及第三方程式庫，例如：[JWT, JSON Web Token](#)  
<[http://www.ruanyifeng.com/blog/2018/07/json\\_web\\_token-tutorial.html](http://www.ruanyifeng.com/blog/2018/07/json_web_token-tutorial.html)>  
參考：[ASP.NET 使用 JWT 進行 WebAPI 驗證 by 全端開發人員天梯](#) <<https://wellwind.idv.tw/blog/2016/11/28/jwt-auth-web-api/>>
4. 為求簡單範例所有邏輯都大鍋炒放在同一個專案裡，實務上則常依性質功能拆成多顆 DLL，有助於分工開發、重複使用、部署換版... 等，好處很多。

That's all folks.



