

# Web API 2

## 技術分享

2016/07/19

網技二課

# 大綱

---

- RESTful API
- 路由機制
- 模型繫結
- IHttpActionResult
- HttpResponseMessage
- 例外處理
- 序列化訊息
- Message Handlers
- 生命週期
- Postman 好用技巧

# RESTful API

# RESTful API

---

Roy Fielding

在 2000 年的博士論文中所提出

- HTTP 規格 (specification) 的主要編撰者
- Apache HTTP Server 共同創辦人

# RESTful API

---

不是一種標準，

而是提供一堆軟體架構設計上的限制

一致性的介面 (Uniform Interface)

透過一致的操作介面 (API)

提高 Client/Server 之間互動的可見性

分層設計 (Layered System)

透過分層設計，讓不同層級的元件可以分工  
或透過負載 平衡提高延展性

# RESTful API

---

## 主從式架構 (Client-Server)

由客戶端單方面發起，表現為 Request/Response 的形式

## 無狀態 (Stateless)

由客戶端負責所有狀態保存

## 可快取特性 (Cacheable)

回應的內容可以在呼叫的過程中適度快取，以改善執行效率

**HTTP** 就是這樣 (同一個人設計的)

# RESTful API

---

一個 RESTful Web API 的操作介面

**主 詞**：網址

**動 詞**：操作 ( GET, POST, PUT )

**內 容**：資源格式 ( JSON, Text, XML )

GET /api/books

{ id: 1, name: "MVC" }

Content-Type: application/json

# RESTful API 建議

---

善用 HTTP 動詞 (Verbs)

善用 HTTP 狀態碼 (Status Code) 表達狀態

使用 SSL 加密連線

擁有良好的 APIs 文件

提供查詢、排序、篩選功能

使用 JSON 回應訊息

參考來源

<http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api>



# 常見的 HTTP 動詞

---

GET	取得訊息
POST	建立訊息
PUT	更新 <b>完整</b> 訊息
PATCH	更新 <b>部分</b> 訊息
DELETE	刪除訊息

# RESTful 風格

---

將動詞設計在網址列中

( 這不能算是 RESTful 風格 )

GET /api/**get**file

POST /api/**upload**file

POST /api/**delete**file

回應訊息時狀態寫在訊息中

( 這也不能算是 RESTful 風格 )

```
{  
  status: "OK",  
  message: "Hello World!"  
}
```

# RESTful 的 API 設計

---

使用一致性的操作介面表達「資源」

- /api/file

將動詞設計在 HTTP Verbs 中

- GET /api/file

- POST /api/file

回應訊息時的狀態寫在 HTTP 狀態碼中

HTTP/1.1 201 Created

Content-Type: text/json; charset=UTF-8

```
{  
  message: "Hello World!"  
}
```

# 常見 HTTP 狀態碼與主要分類

---

## 2xx - 成功 (OK)

- 200 : 成功
- 201 : 資源已建立
- 204 : 處理完成但無回傳資訊

## • 3xx - 重新導向 (Redirection)

- 301 : 永久轉址
- 302 : 暫時轉址
- 304 : 未修改

## • 4xx - 用戶端錯誤 (Client Error)

- 400 : 錯誤的請求
- 401 : 無權限存取
- 404 : 找不到資源
- 409 : 請求的處理發生衝突

## • 5xx - 伺服器錯誤 (Server Error)

- 500 : 伺服器發生錯誤

## 參考資料

<http://blog.miniasp.com/post/2009/01/16/Web-developer-should-know-about-HTTP-Status-Code.aspx>

# RESTful API

---

功能	URI	HTTP Method
新增	/books	POST
刪除	/books/{id}	DELETE
修改	/books/{id}	PUT
查詢	/books/{id}	GET
列表	/books	GET

# 好看好用的 API 文件

---

Swashbuckle 5.0 ( swagger )

- 快速產生文件
- 快速產生 API 測試功能
- 快速使用

ASP.NET Web API 文件產生器 - 使用 Swagger

<http://kevintsengtw.blogspot.tw/2015/12/aspnet-web-api-swagger.html>

# 好看好用的 API 文件

 swagger<http://notice-t.evertrust.com.tw:80/swagger/docs/v1>[Explore](#)

## YCNoticeService.Application

### Header

Parameter	Value	Description
apikey	<input type="text" value="(required)"/>	使用服務的金鑰(需提出申請)

### Notice

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

GET	/apiVersion/Notice/List	取通知列表
DELETE	/apiVersion/Notice	刪除通知
GET	/apiVersion/Notice	通知詳細資料
PATCH	/apiVersion/Notice	更新通知狀態
POST	/apiVersion/Notice	新增通知
GET	/apiVersion/Notice/Summary	通知總覽
GET	/apiVersion/Notice/NewArrive	取得新進訊息
PATCH	/apiVersion/Notice/NewArrive	更新取得新進訊息時間

[ BASE URL: , API VERSION: v1 ]

[ERROR](#) 

# 好看好用的 API 文件

parameter.recipientID  通知對象識別值 query string

Try it out!

[Hide Response](#)

## Curl

```
curl -X GET --header 'Accept: application/json' 'http://notice-t.evertrust.com.tw/v1/Notice/Summary?parameter.recipientType=YCMB&parameter.recipientID=C59515BD-ECA7-4491-4030-00000000-00000000'
```

## Request URL

```
http://notice-t.evertrust.com.tw/v1/Notice/Summary?parameter.recipientType=YCMB&parameter.recipientID=C59515BD-ECA7-4491-4030-00000000-00000000
```

## Response Body

```
{
  "apiVersion": "v1",
  "method": "NoticeSummary.GET",
  "status": "ERROR",
  "id": "556fbb7c-bcfc-4b47-9247-c51bd0e8e608",
  "error": {
    "domain": "NoticeService",
    "code": 403,
    "message": "目前暫時無法提供服務，請稍後再試！",
    "description": "無使用系統權限，請與系統管理員聯絡 !"
  }
}
```



# 路由機制



# 屬性路由 (Attribute Routing)

傳統路由	屬性路由
所有路由交由 <code>WebApiConfig.cs</code> 集中管理	所有路由定義在控制器中
由單一檔案宣告所有控制器的路由規則	由各控制器自行宣告路由規則
難以設計 RESTful APIs 彈性的網址結構	清楚明瞭、結構彈性的路由規則定義

# 屬性路由 (Attribute Routing)

HTTP 動詞	URI 路徑	動作方法
GET	api/products/4	<b>Get</b> Product
DELETE	api/products/4	<b>Delete</b> Product
POST	api/products	<b>Post</b> Product

```
public IActionResult GetProduct(int id)...
```

```
[Route("add")]
```

0 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
public IActionResult PostProduct(Product product)...
```

```
[Route("update/{id:int}")]
```

0 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
public IActionResult PatchProduct(int id, Product product)...
```

```
[Route("delete/{id:int}")]
```

0 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
public IActionResult DeleteProduct(int id)...
```

# 屬性路由

```
[RoutePrefix("products")]
```

1 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
public class ProductsController : ApiController
```

```
{
```

```
    //GET: products
```

```
    [Route("")]
```

0 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
    public IQueryable<Product> GetProduct()...
```

```
    // GET: products/search/Jack
```

```
    [Route("search/{name}")]
```

0 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
    public IHttpActionResult GetSearchProduct(string name)...
```

```
    // GET: products/5
```

```
    [Route("{id:int}")]
```

0 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
    public IHttpActionResult GetProduct(int id)...
```

```
    // POST api/test/5 (用 ~ 可以覆寫 RoutePrefix 設定)
```

```
    [Route("~/api/test/{id:int}")]
```

0 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
    public IHttpActionResult GetProduct(int id)...
```

# 屬性路由

```
[Route("{id:int=87}")]
```

0 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
public IActionResult GetProduct(int id)...
```

上面跟下面 是一樣的效果

```
[Route("{id:int?}")]
```

0 個參考 | 林楊森(網技二課), 4 天前 | 1 位作者, 1 項變更

```
public IActionResult GetProduct(int id = 87)...
```

Int? 就是 nullable  
也就是可以不傳

# HTTP 動作過濾器

支援的 HTTP 動詞

[HttpGet]            [HttpPost]            [HttpPut]  
[HttpDelete]        [HttpPatch]

[AcceptVerbs("Login")]

支援非標準的 HTTP 動詞

```
[HttpGet]  
[HttpPost]  
0 個參考 | 0 項變更 | 0 位作者, 0 項變更  
public IActionResult Product(int id)...
```

但不能算是 RESTful 風格

# 路由參數限定詞

## • 字串限定詞

限定詞	說明	範例
alpha	比對大寫或小寫拉丁字母 (a-z, A-Z)	<code>{x:alpha}</code>
length	比對字串長度範圍	<code>{x:length(6)}</code> <code>{x:length(1,20)}</code>
maxlength	比對字串最大長度	<code>{x:maxlength(10)}</code>
minlength	比對字串最小長度	<code>{x:minlength(10)}</code>

## • 正則表達式限定詞

限定詞	說明	範例
regex	比對字串必須符合一個正規表示式	<code>{x:regex(^\\d{3}-\\d{3}-\\d{4}\$)}</code>

# 路由參數限定詞

限定詞	說明	範例
bool	比對布林值	{x:bool}
datetime	比對 <b>DateTime</b> 日期格式	{x:datetime}
decimal	比對 decimal 數值格式	{x:decimal}
Double	比對 64-bit 浮點數 (floating-point)	{x:double}
float	比對 32-bit 浮點數 (floating-point)	{x:float}
guid	比對 GUID 值	{x:guid}
int	比對 32-bit 整數值 (integer value)	{x:int}
long	比對 64-bit 整數值 (integer value)	{x:long}
max	比對整數的最大值 (必須小於等於一個整數)	{x:max(10)}
min	比對整數的最小值 (必須大於等於一個整數)	{x:min(10)}
range	比對一個整數範圍	{x:range(10,50)}



# 路由名稱

```
[Route("{id}", Name = "GetProductById")]
```

0 個參考 | 0 項變更 | 0 位作者, 0 項變更

```
public IActionResult GetUrl(int id)
```

```
{
```

```
    var uri = Url.Link("GetProductById", new { id = id });
```

```
    return Ok(uri);
```

```
}
```

# 路由參數限定詞

```
[Route("~/v1/Now/{x:datetime}", Name = "GetNow", Order = 1)]  
0 個參考 | 0 項變更 | 0 位作者, 0 項變更  
public IActionResult GetNow(DateTime x)  
{  
    var uri = Url.Link("GetNow", new { x = DateTime.Now });  
    return Ok(uri);  
}
```

GET  Params

Body Cookies (14) Headers (7) Tests Status: 200 OK Time: 25 ms

Pretty Raw Preview JSON

```
1 "http://localhost:44205/v1/Now/01/25/2017%2015:20:05"
```

GET  Params

Body Cookies (14) Headers (6) Tests Status: 404 Not Found Time: 19 ms

Pretty Raw Preview HTML

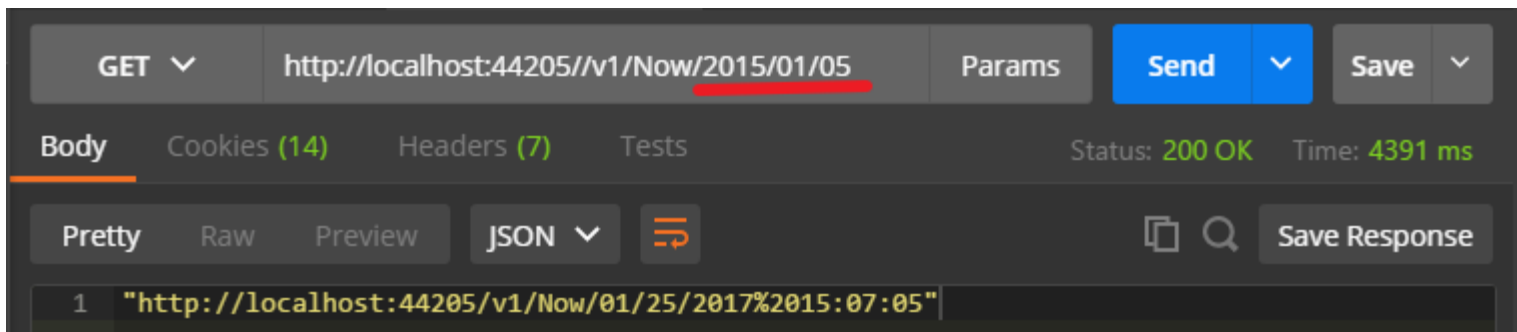
```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR  
  /xhtml1/DTD/xhtml1-strict.dtd">  
2 <html xmlns="http://www.w3.org/1999/xhtml">  
3 <head>
```

# 萬用路由參數

```
[Route("~/v1/Now/{*x:datetime}", Name = "GetNow", Order = 1)]
```




0 個參考 | 0 項變更 | 0 位作者, 0 項變更

```
public IHttpActionResult GetNow(DateTime x)
{
    var uri = Url.Link("GetNow", new { x = DateTime.Now });
    return Ok(uri);
}
```

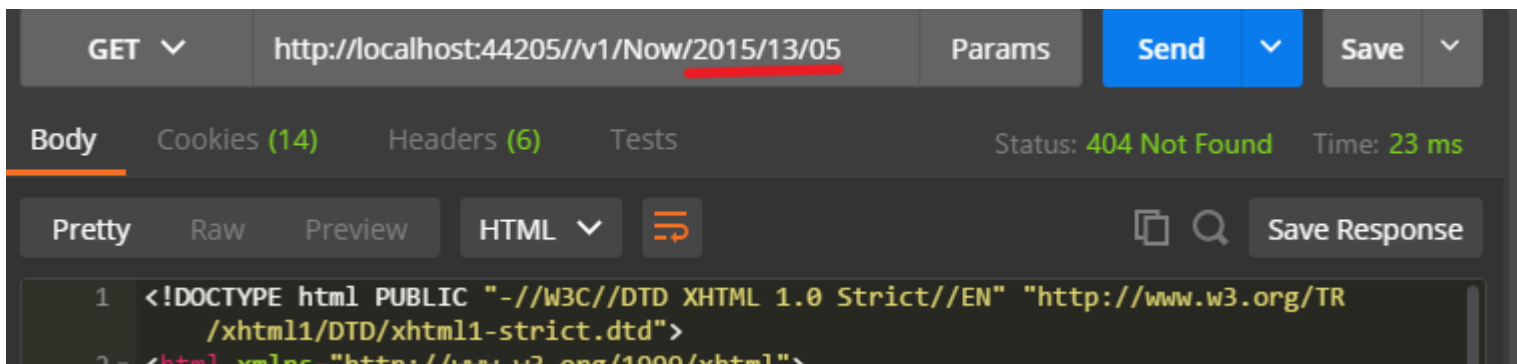


GET ▼ http://localhost:44205/v1/Now/2015/01/05 Params Send ▼ Save ▼

Body Cookies (14) Headers (7) Tests Status: 200 OK Time: 4391 ms




Pretty Raw Preview JSON ▼    Save Response

1 "http://localhost:44205/v1/Now/01/25/2017%2015:07:05"



GET ▼ http://localhost:44205/v1/Now/2015/13/05 Params Send ▼ Save ▼

Body Cookies (14) Headers (6) Tests Status: 404 Not Found Time: 23 ms

Pretty Raw Preview HTML ▼    Save Response

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

2 <html xmlns="http://www.w3.org/1999/xhtml">

# 路由套用順序

---

相同網址結構的路由如何套用？

1. 先比對是否設定路由順序 ( Order ) ( 預設值為 0 )

數字**越小**，優先權**越高**

2. 純字串、無參數、無限制的路由

3. 有路由參數，且有套用限定詞

4. 有路由參數，且**無套用**限定詞

5. 有萬用路由參數 (Wildcard parameter) 且有套用限定詞

6. 有萬用路由參數 (Wildcard parameter) 且**無套用**限定詞

# 請依號碼排列

```
public class OrdersController : ApiController
{
    ① [Route("{id:int}")]
    public HttpResponseMessage Get(int id) { ... }

    ② [Route("details")]
    public HttpResponseMessage GetDetails() { ... }

    ③ [Route("pending", RouteOrder = 1)]
    public HttpResponseMessage GetPending() { ... }

    ④ [Route("{customerName}")]
    public HttpResponseMessage GetByCustomer(string customerName) { ... }

    ⑤ [Route("{*date:datetime}")]
    public HttpResponseMessage Get(DateTime date) { ... }
}
```

# 模型繫結

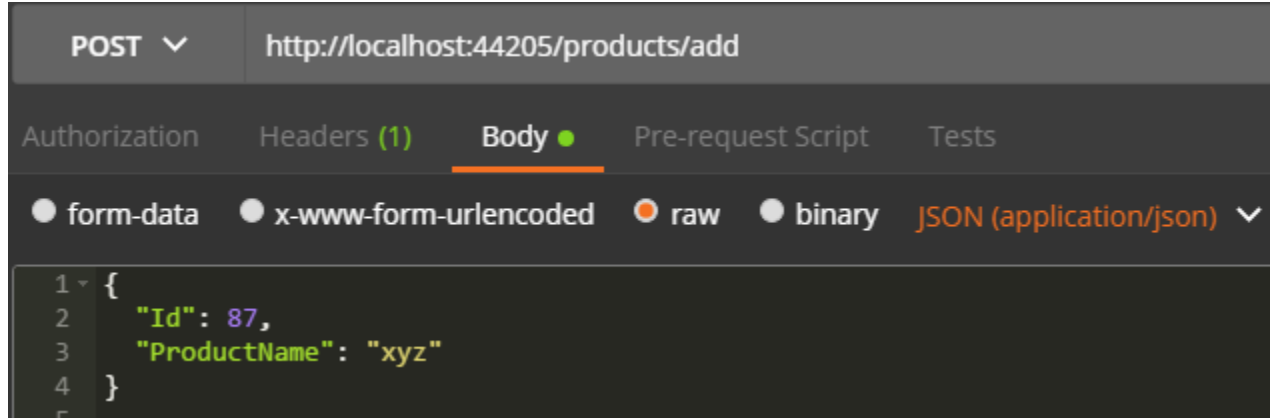
# 簡單模型繫結

預設從 URI 取值 (URI Path 或 Query String)

GET ▾	http://localhost/v1/Now/book/?id=3&name=jack		Params
id		3	≡ ×
name		jack	≡ ×

# 複雜模型繫結

## 預設從 要求主體 (Request Body)





# 簡單模型繫結正常版

```
[Route("~/Demo/{age}/{xx}")]
```

0 個參考 | 0 項變更 | 0 位作者, 0 項變更

```
public IActionResult GetDemo1(int age,string xx,string name)
{
    return Ok(new {age=age,xx=xx,name=name });
}
```

GET ▾

http://localhost:44205/Demo/99/aa?name=jack

Params

Send ▾

Save ▾

Body

Cookies (14)

Headers (7)

Tests

Status: 200 OK Time: 97 ms

Pretty

Raw

Preview

JSON ▾



Save Response

```
1 {
2   "age": 99,
3   "xx": "aa",
4   "name": "jack"
5 }
```



# 簡單模型繫結逆天版 FromUri

```
[Route("~/Demo2/{age}/{xx}")]
0 個參考 | 0 項變更 | 0 位作者, 0 項變更
public IActionResult GetDemo2([FromUri]Demo2 p)
{
    return Ok(new { FromUri = 'Y', age = p.age, xx = p.xx, name = p.name });
}

1 個參考 | 0 項變更 | 0 位作者, 0 項變更
public class Demo2
{
    1 個參考 | 0 項變更 | 0 位作者, 0 項變更
    public int age { get; set; }
    1 個參考 | 0 項變更 | 0 位作者, 0 項變更
    public string xx { get; set; }
    1 個參考 | 0 項變更 | 0 位作者, 0 項變更
    public string name { get; set; }
}
```

GET ▾

http://localhost:44205/Demo2/99/aa?name=jack

Params

Send ▾

Save ▾

Body

Cookies (14)

Headers (7)

Tests

Status: 200 OK

Time: 212 ms

Pretty

Raw

Preview

JSON ▾



Save Response

```
1 {
2   "FromUri": "Y",
3   "age": 99,
4   "xx": "aa",
5   "name": "jack"
6 }
```

# 複雜模型繫結正常版

```
[Route("~/Demo3")]
0 個參考 | 0 項變更 | 0 位作者, 0 項變更
public IActionResult PostDemo3(Demo2 p)
{
    return Ok(new { age = p.age, xx = p.xx, name = $"{p.name}Demo3" });
}
```

POST ▾

http://localhost:44205/Demo3

Params

Send ▾

Save ▾

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

JSON (application/json) ▾

```
1 {
2   "age": 99,
3   "xx": "aa",
4   "name": "jack"
5 }
```

Body

Cookies (14)

Headers (7)

Tests

Status: 200 OK



Time: 4503 ms

Pretty

Raw

Preview

JSON ▾

Save Response

```
1 {
2   "age": 99,
3   "xx": "aa",
4   "name": "jackDemo3"
5 }
```

# 複雜模型繫結逆天版 FromBody

```
[Route("~/Demo4")]
0 個參考 | 0 項變更 | 0 位作者, 0 項變更
public IActionResult PostDemo4([FromBody]int id)
{
    return Ok(new {id=id });
}
```

POST ☐ http://localhost:44205/Demo4 Params Send ☐ Save ☐

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) ☐

1 345

Body Cookies (14) Headers (7) Tests Status: 200 OK Time: 21 ms

Pretty Raw Preview JSON ☐ Save Response

```
1 {
2   "id": 345
3 }
```

每個 Action 最多只能  
有一個參數從 Request Body 取值！

# IHttpActionResult

# Ok

```
[Route("IHttpActionResult/Ok")]  
0 個參考 | 林楊森(網技二課), 15 小時前 | 1 位作者, 1  
public IHttpActionResult GetDemo1()  
{  
    //200  
    return Ok(new { test = "ok" });  
}
```

GET ▼

http://localhost:44205/IHttpActionResult/Ok

Params

Send ▼

Save ▼

Body

Cookies (14)

Headers (7)

Tests

Status: 200 OK Time: 8794 ms

Pretty

Raw

Preview

JSON ▼



Save Response

```
1 {  
2   "test": "ok"  
3 }
```

# BadRequest

```
[Route("IHttpActionResult/BadRequest")]
```

0 個參考 | 林楊森(網技二課), 15 小時前 | 1 位作者, 1 項

```
public IHttpActionResult GetDemo2()
{
    //400
    return BadRequest("Bad Request");
}
```

GET ▼

http://localhost:44205/IHttpActionResult/BadRequest

Params

Send ▼

Save ▼

Body

Cookies (14)

Headers (7)

Tests

Status: 400 Bad Request

Time: 4803 ms

Pretty

Raw

Preview

JSON ▼

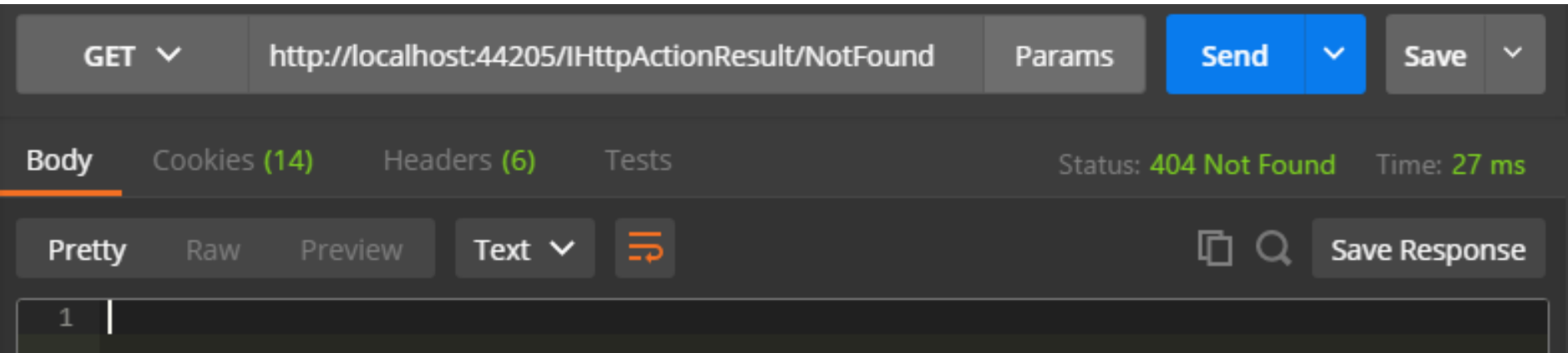


Save Response

```
1 {
2   "Message": "Bad Request"
3 }
```

# NotFound



```
[Route("IHttpActionResult/NotFound")]  
0 個參考 | 林楊森(網技二課), 15 小時前 | 1 位作者,  
public IActionResult GetDemo3()  
{  
    //404  
    return NotFound();  
}
```



The screenshot shows a web browser's developer tools interface. The top bar displays the HTTP method as GET, the URL as http://localhost:44205/IHttpActionResult/NotFound, and the status as 404 Not Found. The 'Body' tab is selected, showing the response content. The 'Text' view is chosen, and the response is empty. The 'Save Response' button is visible.

GET ⌵ http://localhost:44205/IHttpActionResult/NotFound Params Send ⌵ Save ⌵

Body Cookies (14) Headers (6) Tests Status: 404 Not Found Time: 27 ms

Pretty Raw Preview Text ⌵   Save Response

1

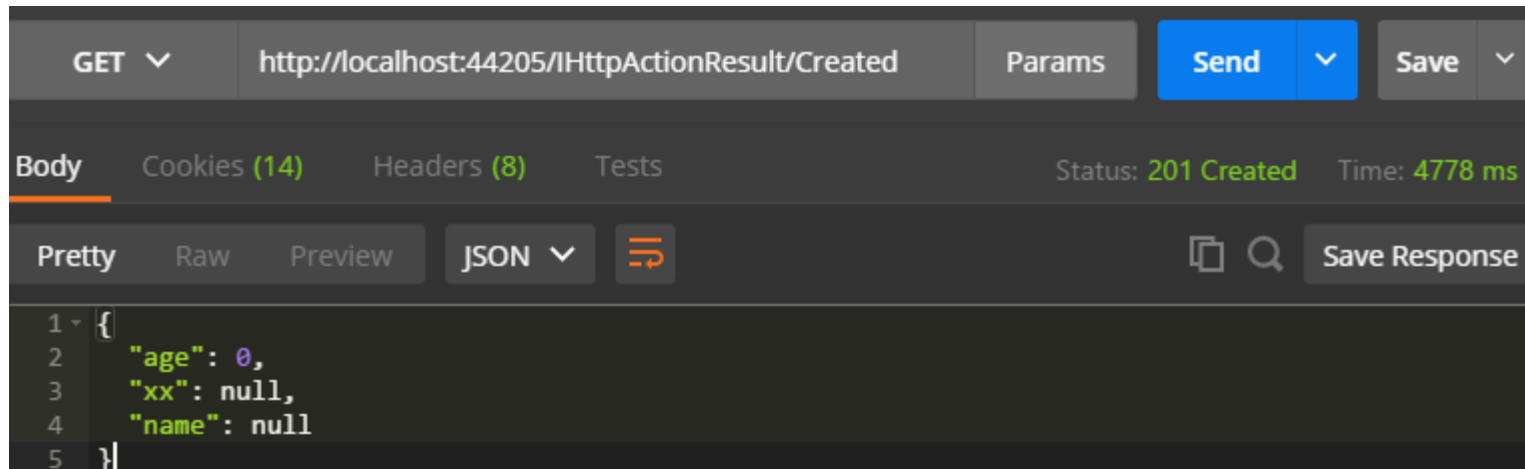


# Created

```
[Route("IHttpActionResult/Created")]
```

0 個參考 | 林楊森(網技二課), 15 小時前 | 1 位作者, 1 項變更

```
public IHttpActionResult GetDemo4()
{
    //201
    return Created<Demo2>("http://test.com/GetDemo4", new Demo2());
}
```



The screenshot shows a web browser's developer tools interface. The top bar displays the method 'GET', the URL 'http://localhost:44205/IHttpActionResult/Created', and buttons for 'Params', 'Send', and 'Save'. Below this, the 'Body' tab is selected, showing the response status '201 Created' and the time '4778 ms'. The response body is displayed in JSON format, showing a successful creation of a resource with fields 'age', 'xx', and 'name'.

```
1 {
2   "age": 0,
3   "xx": null,
4   "name": null
5 }
```

# Created

```
[Route("IHttpActionResult/Created")]
```

0 個參考 | 林楊森(網技二課), 15 小時前 | 1 位作者, 1 項變更

```
public IHttpActionResult GetDemo4()
{
    //201
    return Created<Demo2>("http://test.com/GetDemo4", new Demo2());
}
```

Body Cookies (14) **Headers (8)** Tests

**Cache-Control** → no-cache

**Content-Length** → 31

**Content-Type** → application/json; charset=utf-8

**Date** → Thu, 26 Jan 2017 01:10:57 GMT

**Expires** → -1

**Location** → http://test.com/GetDemo4

**Pragma** → no-cache

# InternalServerError

```
[Route("IHttpActionResult/InternalServerError")]
```

0 個參考 | 林楊森(網技二課), 15 小時前 | 1 位作者, 1 項變更

```
public IHttpActionResult GetDemo5()
```

```
{  
    //500  
    return InternalServerError(new Exception("意外意外"));  
}
```

GET ▾

http://localhost:44205/IHttpActionResult/InternalServerError

Params

Send

▾

Save

▾

Body

Cookies (14)

Headers (7)

Tests

Status: 500 Internal Server Error

Time: 88 ms

Pretty

Raw

Preview

JSON ▾



Save Response

```
1 {  
2   "Message": "發生錯誤。",  
3   "ExceptionMessage": "意外意外",  
4   "ExceptionType": "System.Exception",  
5   "StackTrace": null  
6 }
```

# Content

```
[Route("IHttpActionResult/Content")]
```

0 個參考 | 林楊森(網技二課), 15 小時前 | 1 位作者, 1 項變更

```
public IHttpActionResult GetDemo6()
{
    //404
    return Content(HttpStatusCode.NotFound, "騙你的的啦");
}
```

GET ▾

http://localhost:44205/IHttpActionResult/Content

Params

Send ▾

Save ▾

Body

Cookies (14)

Headers (7)

Tests

Status: 404 Not Found

Time: 41 ms

Pretty

Raw

Preview

JSON ▾



Save Response

1 "騙你的的啦"

# Content

```
[Route("IHttpActionResult/Content2")]
```

0 個參考 | 林楊森(網技二課), 15 小時前 | 1 位作者, 1 項變更

```
public IHttpActionResult GetDemo7()
```

```
{
```

```
    //410
```

```
    return Content<Demo2>(HttpStatusCode.Gone, new Demo2());
```

```
}
```

GET ▾

http://localhost:44205/IHttpActionResult/Content2

Params

Send ▾

Save ▾

Body

Cookies (14)

Headers (7)

Tests

Status: 410 Gone

Time: 38 ms

Pretty

Raw

Preview

JSON ▾



Save Response

```
1 {  
2   "age": 0,  
3   "xx": null,  
4   "name": null  
5 }
```

# HttpResponseMessage

# 最靈活的方式

```
[Route("HttpResponseMessage/Demo1")]
```

0 個參考 | 0 項變更 | 0 位作者, 0 項變更

```
public HttpResponseMessage GetDemo1()
```

```
{
```

```
    var response = Request.CreateResponse<Demo2>(
```

```
        HttpStatusCode.Created,
```

```
        new Demo2(),
```

```
        new JsonMediaTypeFormatter() { Indent = true }, "text/json");
```

```
    response.ReasonPhrase = "YC";
```

```
    return response;
```

```
}
```

GET ▾

http://localhost:44205/HttpResponseMessage/Demo1

Params

Send ▾

Save ▾

Body

Cookies (14)

Headers (7)

Tests

Status: 201 YC

Time: 4719 ms

Pretty

Raw

Preview

Text ▾



Save Response

```
1 {  
2   "age": 0,  
3   "xx": null,  
4   "name": null  
5 }
```

# 例外處理



# 例外處理 - 1

```
[Route("Exception/NotImplementedException")]
0 個參考 | 0 項變更 | 0 位作者, 0 項變更
public IHttpActionResult GetNotImplementedException()
{
    throw new NotImplementedException("This method is not implemented");
}
```

Pretty

Raw

Preview

JSON



```
1 {
2   "Message": "發生錯誤。",
3   "ExceptionMessage": "This method is not implemented",
4   "ExceptionType": "System.NotImplementedException",
5   "StackTrace": "    於 WebAPILab.Controllers.ExceptionDemoController.GetNotImplementedException() 於 D
      :\\setup\\sam\\Lab\\WebAPILab\\WebAPILab\\Controllers\\ExceptionDemoController.cs: 行 15\\r\\n    於 lambda_
      .Controllers.ReflectedHttpActionDescriptor.ActionExecutor.<>c__DisplayClass10.<GetExecutor>b__9(Object ins
      .Controllers.ReflectedHttpActionDescriptor.ActionExecutor.Execute(Object instance, Object[] arguments)\\r\\n
      .ReflectedHttpActionDescriptor.ExecuteAsync(HttpControllerContext controllerContext, IDictionary`2 argumen
      先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n    於 System.Runtime.CompilerServices.TaskAwaiter.ThrowForNo
      .TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\\r\\n    於 System.Web.Http.Controllers.ApiCo
      ()\\r\\n--- 先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n    於 System.Runtime.CompilerServices.TaskAwaiter.
      .CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\\r\\n    於 System.Web.Http.
      ()\\r\\n--- 先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n    於 System.Runtime.CompilerServices.TaskAwaiter.
      .CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\\r\\n    於 System.Web.Http.
      .MoveNext()\\r\\n--- 先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n    於 System.Runtime.CompilerServices.Tas
      .Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\\r\\n    於 System.W
      .MoveNext()"
6 }
```

# ExceptionHandlerAttribute

```
public class NotImplementedExceptionHandlerAttribute : ExceptionHandlerAttribute
{
    0 個參考 | 0 項變更 | 0 位作者, 0 項變更
    public override void OnException(HttpContext context)
    {
        if (context.Exception is NotImplementedException)
        {
            context.Response = new HttpResponseMessage(HttpStatusCode.NotFound);
        }
    }
}
```

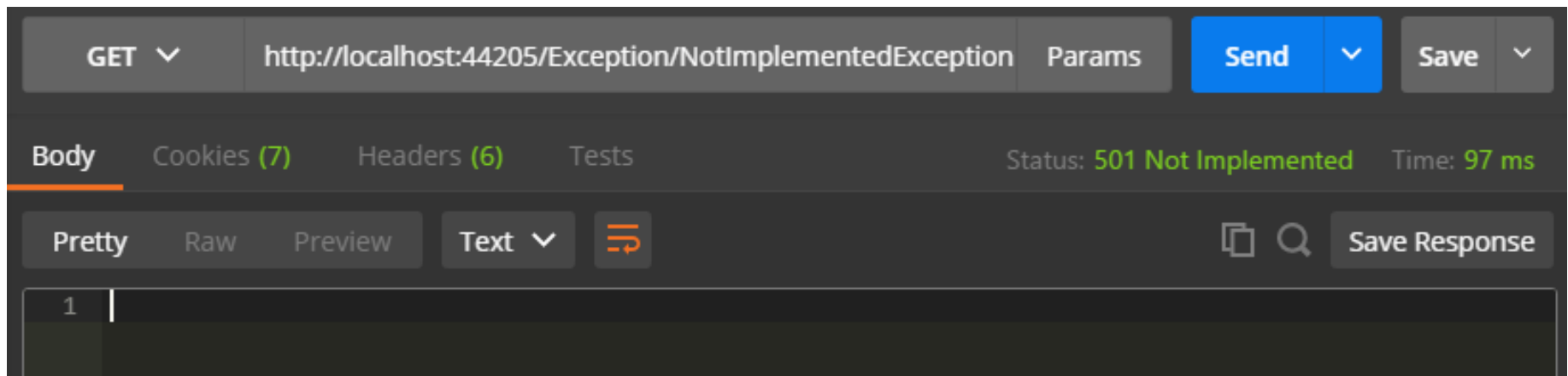
```
public static class WebApiConfig
{
    1 個參考 | 林楊森(網技二課), 16 天前 | 1 位作者, 2 項變更
    public static void Register(HttpConfiguration config)
    {
        //加入自訂ExceptionHandlerAttribute
        GlobalConfiguration.Configuration.Filters.Add(new NotImplementedExceptionHandlerAttribute());
    }
}
```

By action, By controller 請看官方範例

<https://www.asp.net/web-api/overview/error-handling/exception-handling>

# ExceptionHandlerAttribute

```
public class NotImplementedExceptionHandlerAttribute : ExceptionHandlerAttribute
{
    0 個參考 | 0 項變更 | 0 位作者, 0 項變更
    public override void OnException(HttpContext context)
    {
        if (context.Exception is NotImplementedException)
        {
            context.Response = new HttpResponseMessage(HttpStatusCode.NotFound);
        }
    }
}
```



The screenshot shows a web browser interface with a dark theme. The address bar displays the URL `http://localhost:44205/Exception/NotImplementedException` with a GET method selected. To the right of the address bar are buttons for 'Send', 'Save', and a dropdown arrow. Below the address bar, the 'Body' tab is selected, showing a status of '501 Not Implemented' and a response time of '97 ms'. Other tabs like 'Cookies (7)', 'Headers (6)', and 'Tests' are visible. At the bottom, there are buttons for 'Pretty', 'Raw', 'Preview', and 'Text', along with a 'Save Response' button. The main content area is currently empty, showing only a line number '1'.

# 例外處理 - 2

```
[Route("Exception/DivideByZeroException")]
0 個參考 | 0 項變更 | 0 位作者, 0 項變更
public IHttpActionResult GetDivideByZeroException()
{
    int a = 0;
    var b = 1 / a;
    return Ok(b);
}
```

Pretty

Raw

Preview

JSON



```
1 {
2   "Message": "發生錯誤。",
3   "ExceptionMessage": "嘗試以零除。",
4   "ExceptionType": "System.DivideByZeroException",
5   "StackTrace": "    於 WebAPILab.Controllers.ExceptionDemoController.GetDivideByZeroException() 於 D
      :\\setup\\sam\\Lab\\WebAPILab\\WebAPILab\\Controllers\\ExceptionDemoController.cs: 行 22\\r\\n 於
      .Controllers.ReflectedHttpActionDescriptor.ActionExecutor.<>c__DisplayClass10.<GetExecutor>b__9(
      .Controllers.ReflectedHttpActionDescriptor.ActionExecutor.Execute(Object instance, Object[] argu
      .ReflectedHttpActionDescriptor.ExecuteAsync(HttpControllerContext controllerContext, IDictionary
      先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n 於 System.Runtime.CompilerServices.TaskAwaiter.
      .TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\\r\\n 於 System.Web.Http.Control
      ()\\r\\n--- 先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n 於 System.Runtime.CompilerServices.Ta
      .CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\\r\\n 於 System
      ()\\r\\n--- 先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n 於 System.Runtime.CompilerServices.Ta
      .CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\\r\\n 於 System
      .MoveNext()\\r\\n--- 先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n 於 System.Runtime.CompilerSe
      .Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\\r\\n 於
      .MoveNext()\\r\\n--- 先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n 於 System.Web.Http.Controlle
      先前擲回例外狀況之位置中的堆疊追蹤結尾 ---\\r\\n 於 System.Runtime.CompilerServices.TaskAwaiter.
      .TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\\r\\n 於 System.Web.Http.Dispatc
6 }
```

# ExceptionHandler

```
public class OopsExceptionHandler : ExceptionHandler
{
    0 個參考 | 林楊森(網技二課), 16 天前 | 1 位作者, 1 項變更
    public override void Handle(ExceptionHandlerContext context)
    {
        var exception = context.Exception;
        context.Result = new TextPlainErrorResult(context.ExceptionContext.Request, exception);
    }

    2 個參考 | 林楊森(網技二課), 16 天前 | 1 位作者, 1 項變更
    public class TextPlainErrorResult : IHttpActionResult
    {
        1 個參考 | 林楊森(網技二課), 16 天前 | 1 位作者, 1 項變更
        public Task<HttpResponseMessage> ExecuteAsync(CancellationTokens cancellationTokens)
        {
            var errorContent = new JObject()
            {
                new JProperty("status", "ERROR" ),
                new JProperty("message", Exception.Message)
            };

            var response = this.Request.CreateResponse(
                HttpStatusCode.InternalServerError, errorContent);
            return Task.FromResult(response);
        }
    }
}
```

# ExceptionHandler

```
public static class WebApiConfig
{
    1 個參考 | 林楊森(網技二課), 16 天前 | 1 位作者, 2 項變更
    public static void Register(HttpConfiguration config)
    {
        //Global Error Handling
        config.Services.Replace(typeof(IExceptionHandler), new OopsExceptionHandler());

        //加入自訂ExceptionFilterAttribute
        GlobalConfiguration.Configuration.Filters.Add(new NotImplExceptionFilterAttribute());
    }
}
```

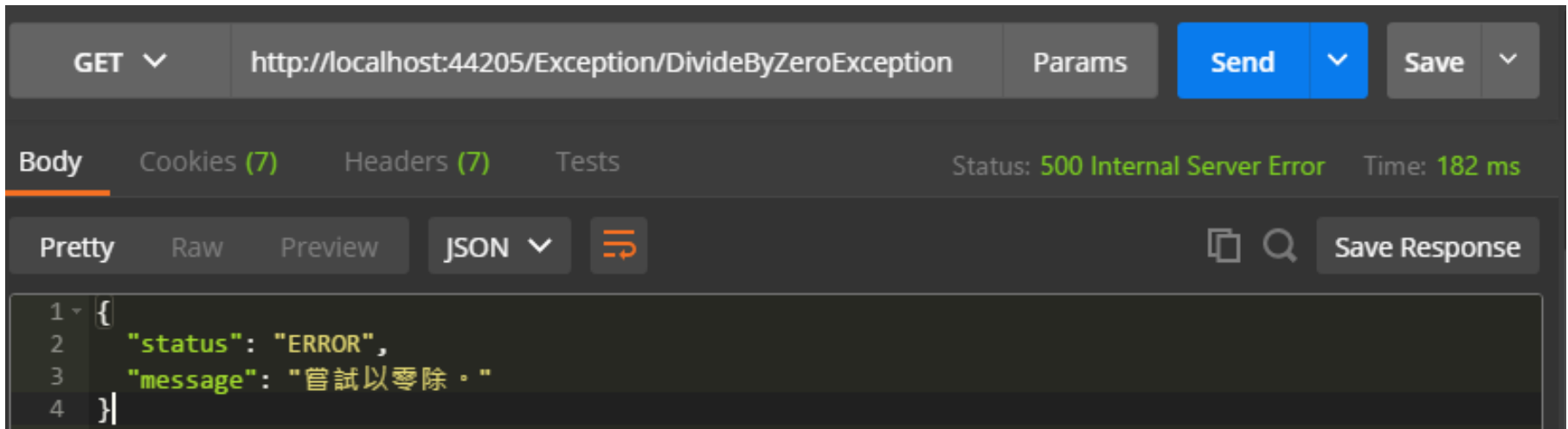
官方範例

<https://www.asp.net/web-api/overview/error-handling/web-api-global-error-handling>

# ExceptionHandler

```
var errorContent = new JObject()
{
    new JProperty("status", "ERROR" ),
    new JProperty("message", Exception.Message)
};

var response = this.Request.CreateResponse(
    HttpStatusCode.InternalServerError, errorContent);
return Task.FromResult(response);
```



The screenshot shows a web browser's developer tools interface. The top bar displays the method 'GET', the URL 'http://localhost:44205/Exception/DivideByZeroException', and buttons for 'Params', 'Send', and 'Save'. Below this, the 'Body' tab is selected, showing the response status '500 Internal Server Error' and time '182 ms'. The response body is displayed in JSON format, showing an error with status 'ERROR' and message '嘗試以零除。'.

GET  Params Send Save

Body Cookies (7) Headers (7) Tests Status: 500 Internal Server Error Time: 182 ms

Pretty Raw Preview JSON

```
1 {
2   "status": "ERROR",
3   "message": "嘗試以零除。"
4 }
```

# ExceptionHandler

```
//Global Error Handling
config.Services.Replace(typeof(ExceptionHandler), new OopsExceptionHandler());

//加入自訂ExceptionFilterAttribute
//GlobalConfiguration.Configuration.Filters.Add(new NotImplExceptionFilterAttribute());
```

GET ▾

http://localhost:44205/Exception/NotImplementedException

Params

Send

▾

Save

▾

Body

Cookies (7)

Headers (7)

Tests

Status: 500 Internal Server Error

Time: 35 ms

Pretty

Raw

Preview

JSON ▾



Save Response

```
1 {
2   "status": "ERROR",
3   "message": "This method is not implemented"
4 }
```

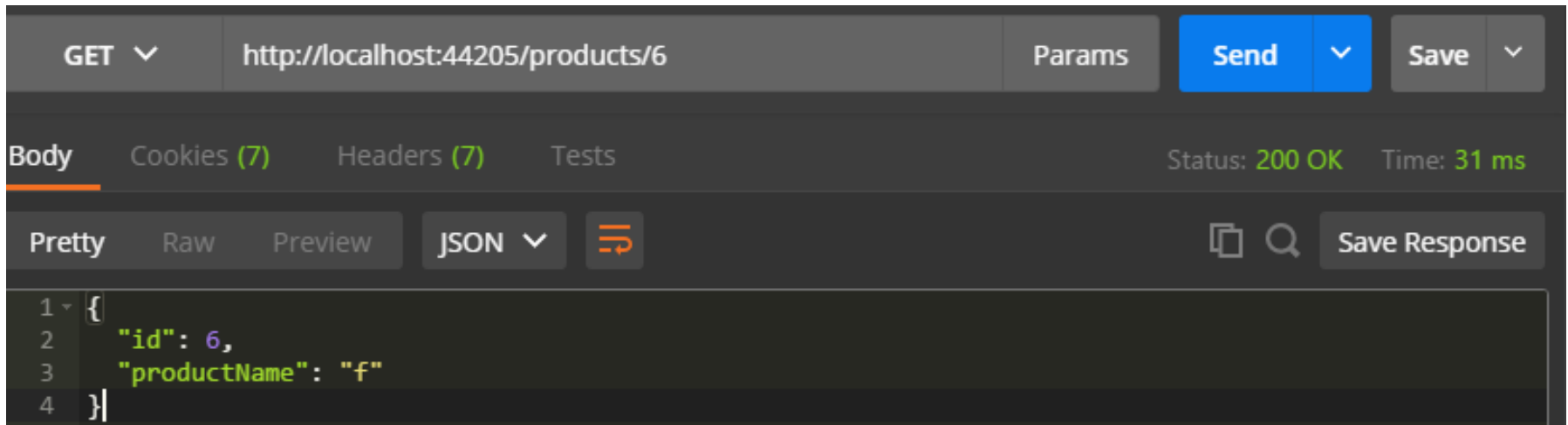


# 序列化訊息

# JSON only

```
//移除XML Formatter
GlobalConfiguration.Configuration.Formatters.XmlFormatter.SupportedMediaTypes.Clear();

//取得 JsonFormatter
var json = GlobalConfiguration.Configuration.Formatters.JsonFormatter;
//輸出 Camel Casing 格式 (不改變 Model 定義)
json.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();
//Json.NET 預設會序列化為本地時間，也可改成預設序列化為 UTC 時間
json.SerializerSettings.DateTimeZoneHandling = Newtonsoft.Json.DateTimeZoneHandling.Utc;
```



GET  Params

Body Cookies (7) Headers (7) Tests Status: 200 OK Time: 31 ms

Pretty Raw Preview JSON

```
1 {
2   "id": 6,
3   "productName": "f"
4 }
```

# Message Handlers



# 常用於紀錄 Log

```
internal class SampleHandler : DelegatingHandler
{
    private string _text;
    private string _indentation;

    2 個參考 | 0 項變更 | 0 位作者 · 0 項變更
    public SampleHandler(string text, int indent)
    {
        if (text == null){throw new ArgumentNullException("text");}
        if (indent < 0){throw new ArgumentException("indent");}
        _text = text;
        _indentation = new String(' ', indent);
    }

    1 個參考 | 0 項變更 | 0 位作者 · 0 項變更
    protected override async Task<HttpResponseMessage> SendAsync(HttpRequestMessage request, Canc
    {
        Debug.WriteLine("{0}{1} request path", _indentation, _text);

        HttpResponseMessage response = await base.SendAsync(request, cancellationToken);

        Debug.WriteLine("{0}{1} response path", _indentation, _text);

        return response;
    }
}
```

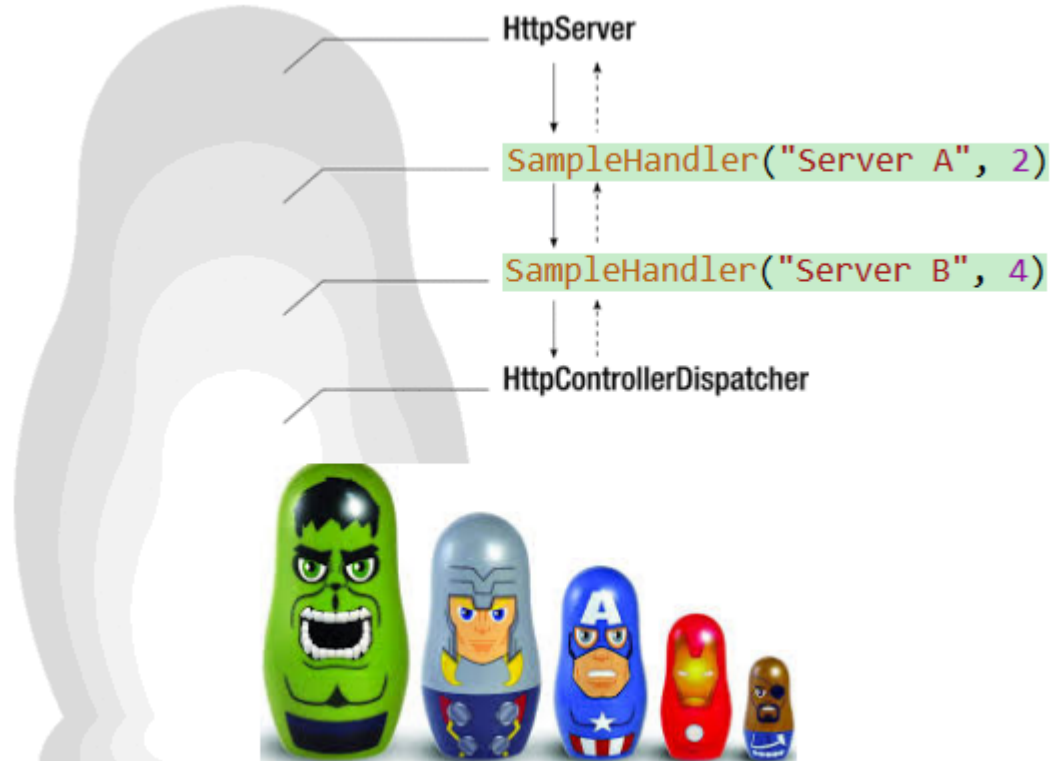
```
public static void Register(HttpConfiguration config)
{
    config.MessageHandlers.Add(new SampleHandler("Server A", 2));
    config.MessageHandlers.Add(new SampleHandler("Server B", 4));
}
```

# 常用於紀錄 Log

## 輸出

顯示輸出來源(S): 偵錯

```
02:13.513    Server A request path
02:13.513    Server B request path
02:13.513    Server B response path
02:13.513    Server A response path
```



實務上使用 HFBuyService - 好房網買屋 APP 後端 WebApi

<http://github.evertrust.com.tw/g0031/HFBuyService/src/master/HouseFun.AppBuy.ClientApplication/Infrastructure/MessageHandlers/LogMessageHandler.cs>

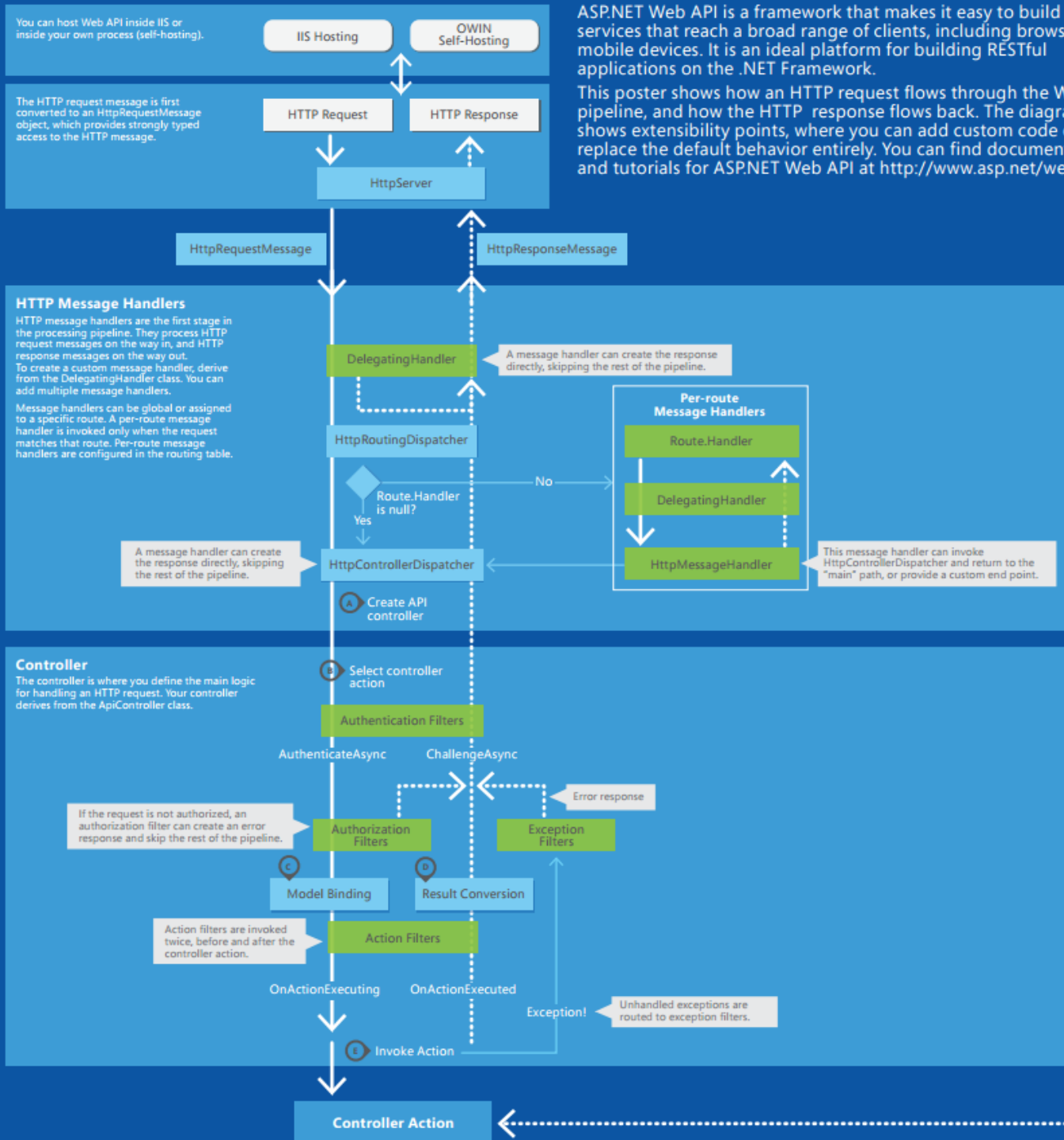
# 該來的逃不了 - 生命週期

The ASP.NET Web API 2 HTTP Message Lifecycle in 43 Easy Steps

<https://www.exceptionnotfound.net/the-asp-net-web-api-2-http-message-lifecycle-in-43-easy-steps-2/>

ASP.NET Web API is a framework that makes it easy to build services that reach a broad range of clients, including browsers and mobile devices. It is an ideal platform for building RESTful applications on the .NET Framework.

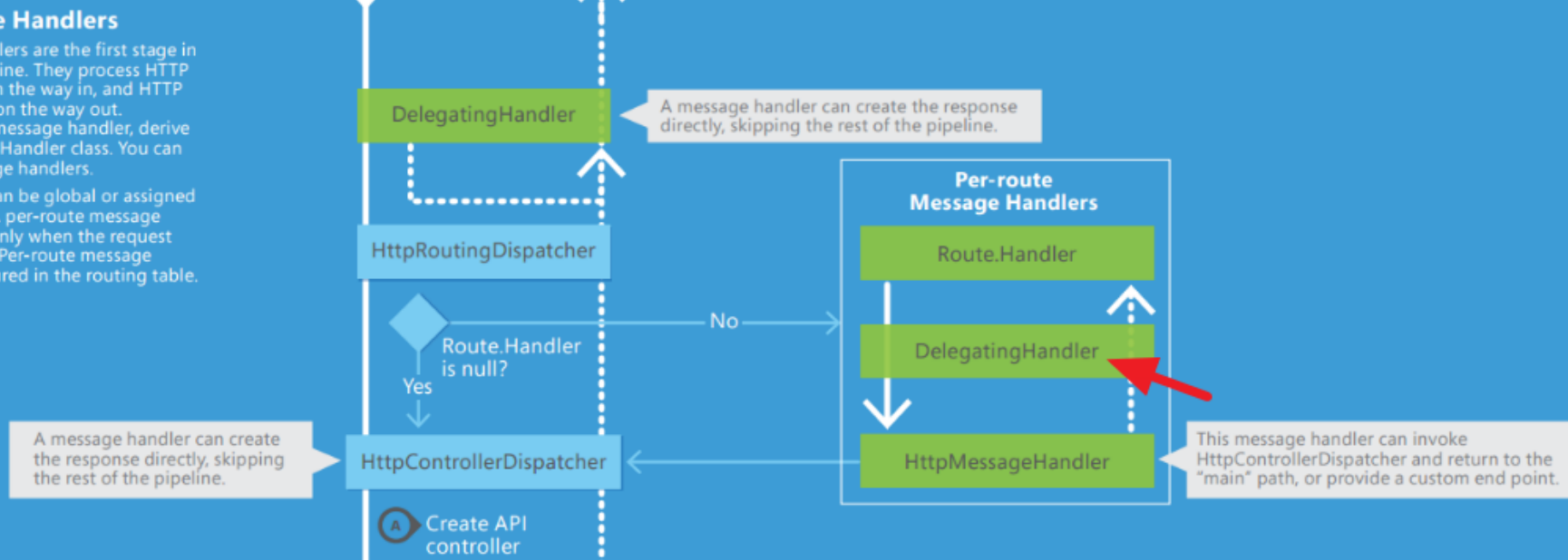
This poster shows how an HTTP request flows through the Web API pipeline, and how the HTTP response flows back. The diagram shows extensibility points, where you can add custom code to replace the default behavior entirely. You can find documents and tutorials for ASP.NET Web API at <http://www.asp.net/webapi>.



## HTTP Message Handlers

HTTP message handlers are the first stage in the processing pipeline. They process HTTP request messages on the way in, and HTTP response messages on the way out. To create a custom message handler, derive from the `DelegatingHandler` class. You can add multiple message handlers.

Message handlers can be global or assigned to a specific route. A per-route message handler is invoked only when the request matches that route. Per-route message handlers are configured in the routing table.



```
internal class SampleHandler : DelegatingHandler
```

```
{  
    1 個參考 | 0 項變更 | 0 位作者 · 0 項變更  
    protected override async Task<HttpResponseMessage> SendAsync(HttpRequestMessage request,  
    {  
        Debug.WriteLine("{0}{1} request path", _indentation, _text);  
  
        HttpResponseMessage response = await base.SendAsync(request, cancellationToken);  
  
        Debug.WriteLine("{0}{1} response path", _indentation, _text);  
  
        return response;  
    }  
}
```

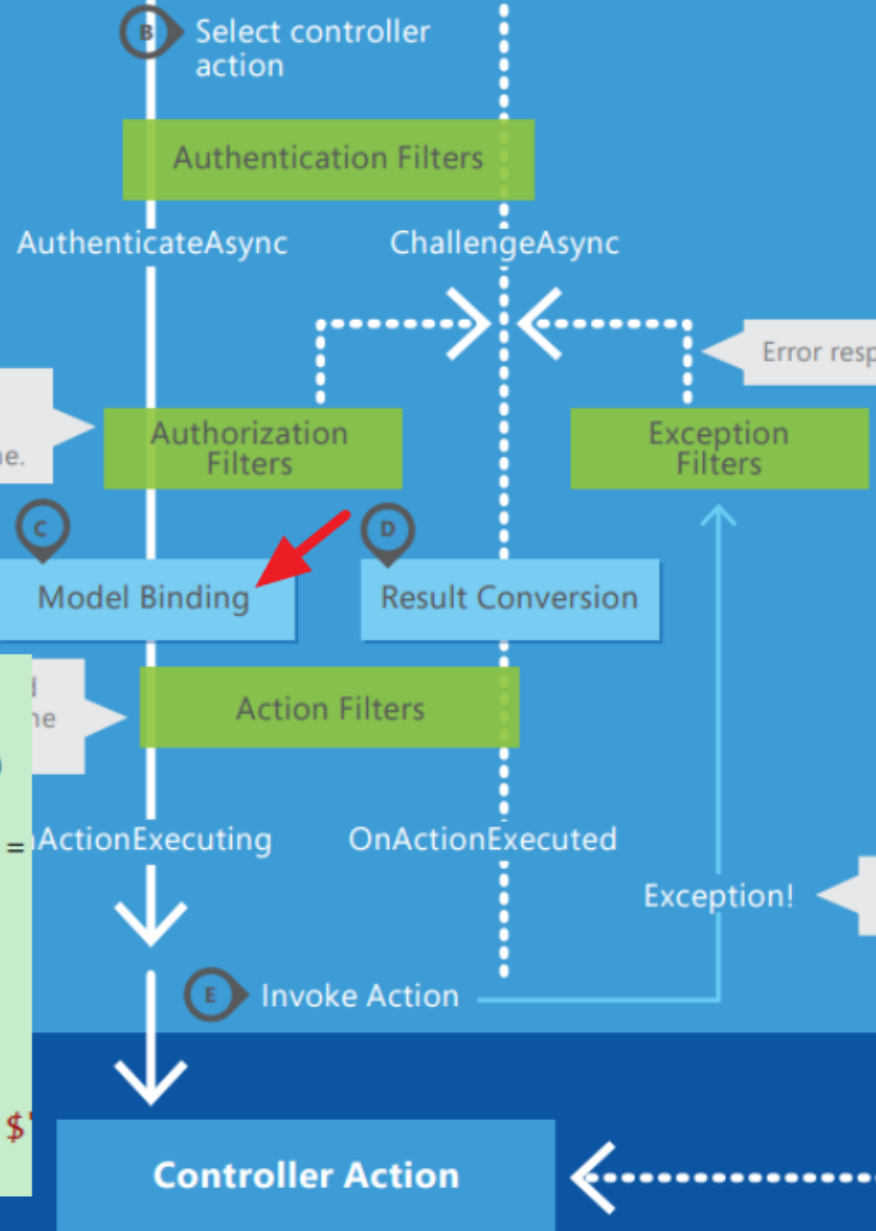


## Controller

The controller is where you define the main logic for handling an HTTP request. Your controller derives from the ApiController class.

```
[Route("~/Demo2/{age}/{xx}")]
0 個參考 | 林楊森(網技二課), 11 天前 | 1 位作者 · 1 項變更
public IHttpActionResult GetDemo2([FromUri] Demo2 p)
{
    return Ok(new { FromUri = 'Y', age = p.age, xx = p.xx });
}

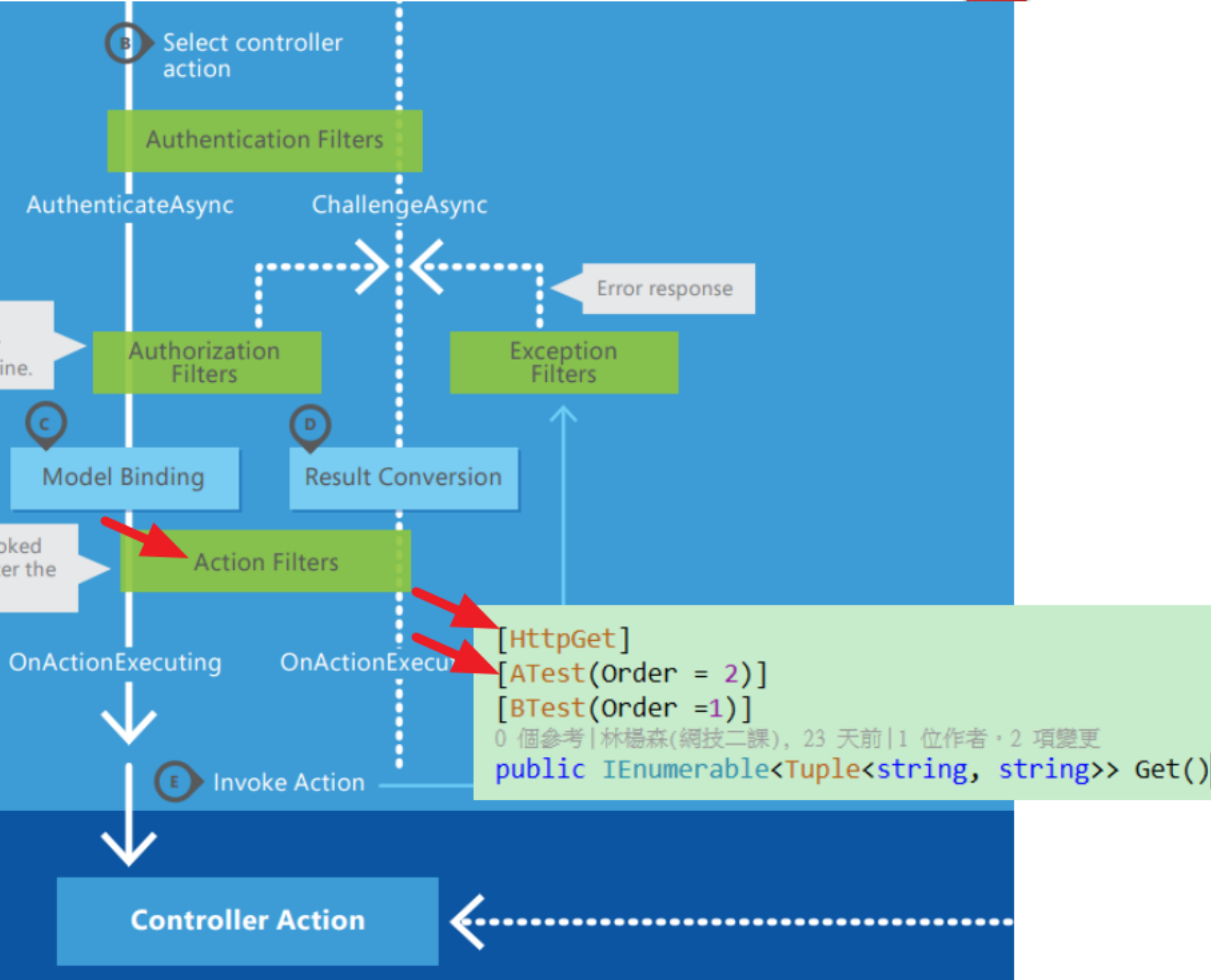
[Route("~/Demo3")]
0 個參考 | 林楊森(網技二課), 11 天前 | 1 位作者 · 1 項變更
public IHttpActionResult PostDemo3(Demo2 p)
{
    return Ok(new { age = p.age, xx = p.xx, name = $"Demo3 {p.age}" });
}
```



the main logic  
controller  
5.

authorized, an  
create an error  
rest of the pipeline.

on filters are invoked  
e, before and after the  
roller action.



If the request is not authorized, an authorization filter can create an error response and skip the rest of the pipeline.

Authorization Filters

Exception Filters

Model Binding

Result Conversion

Action filters are invoked twice, before and after the controller action.

Action Filters

OnActionExecuting

OnActionExecuted

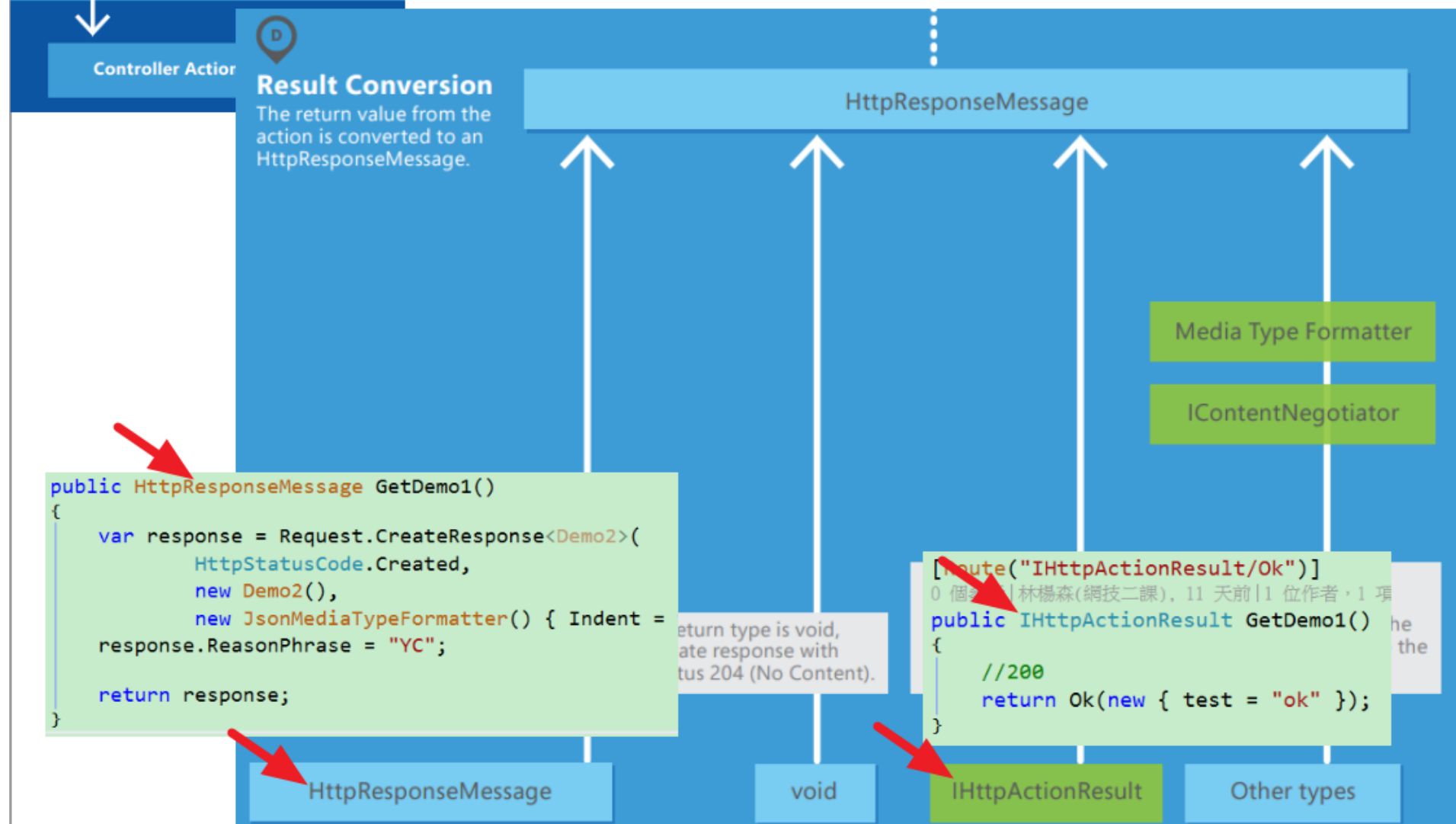
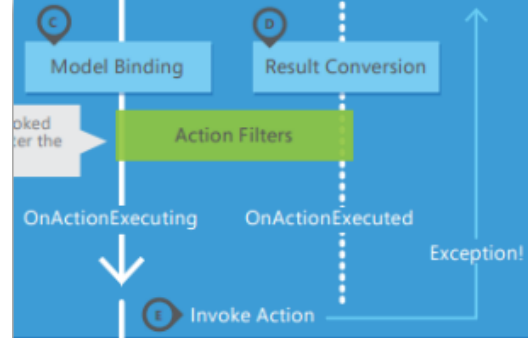
Exception!

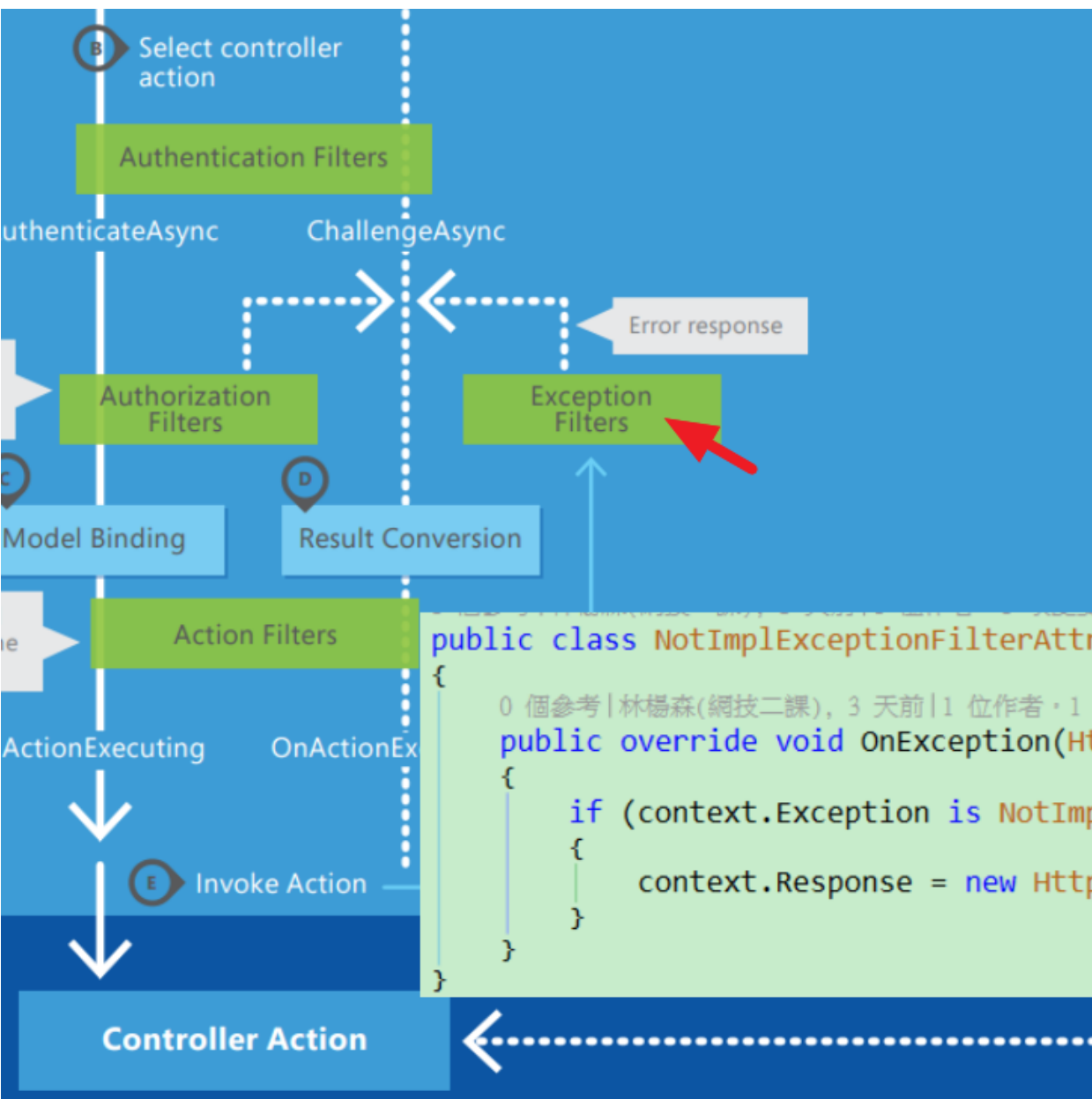
Unhandled exceptions are routed to exception filters.

Invoke Action

Controller Action

```
public IActionResult GetDivideByZeroException()
{
    int a = 0;
    var b = 1 / a;
    return Ok(b);
}
```

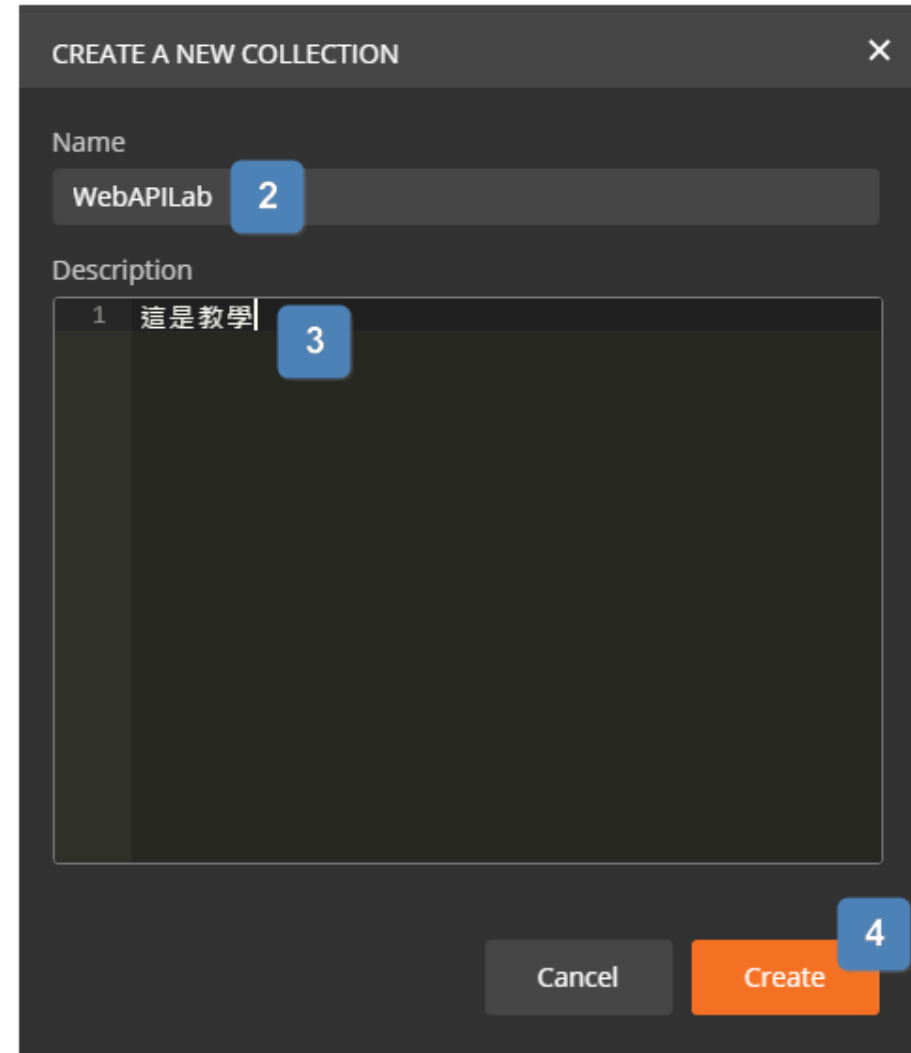
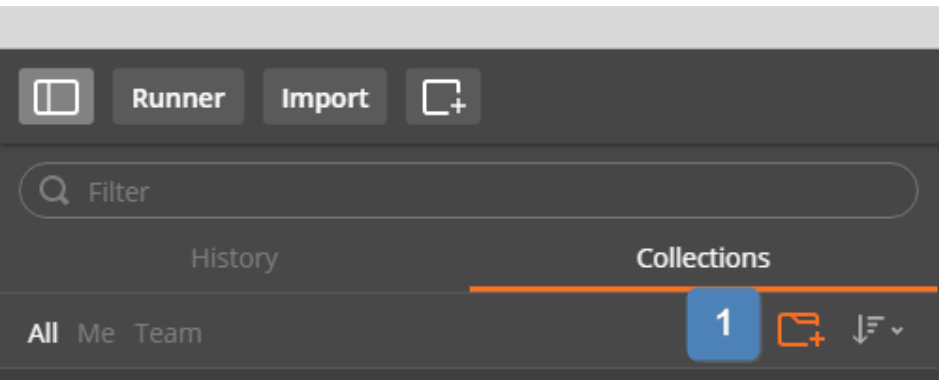




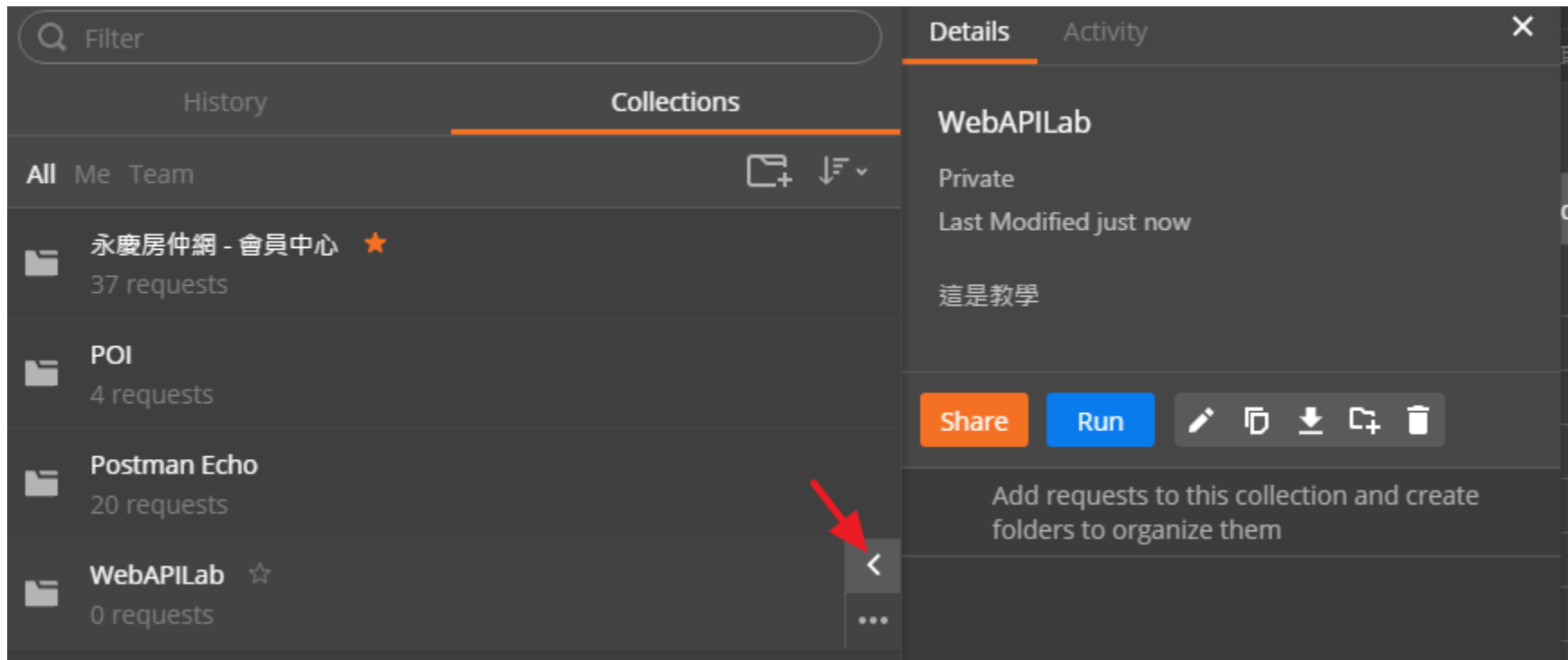
```
public class NotImplExceptionFilterAttribute : ExceptionFilterAttribute
{
    0 個參考 | 林楊森(網技二課), 3 天前 | 1 位作者 · 1 項變更
    public override void OnException(HttpContext context)
    {
        if (context.Exception is NotImplementedException)
        {
            context.Response = new HttpResponseMessage(HttpStatusCode.NotImplemented);
        }
    }
}
```

# Postman 好用技巧

# 建立資料夾

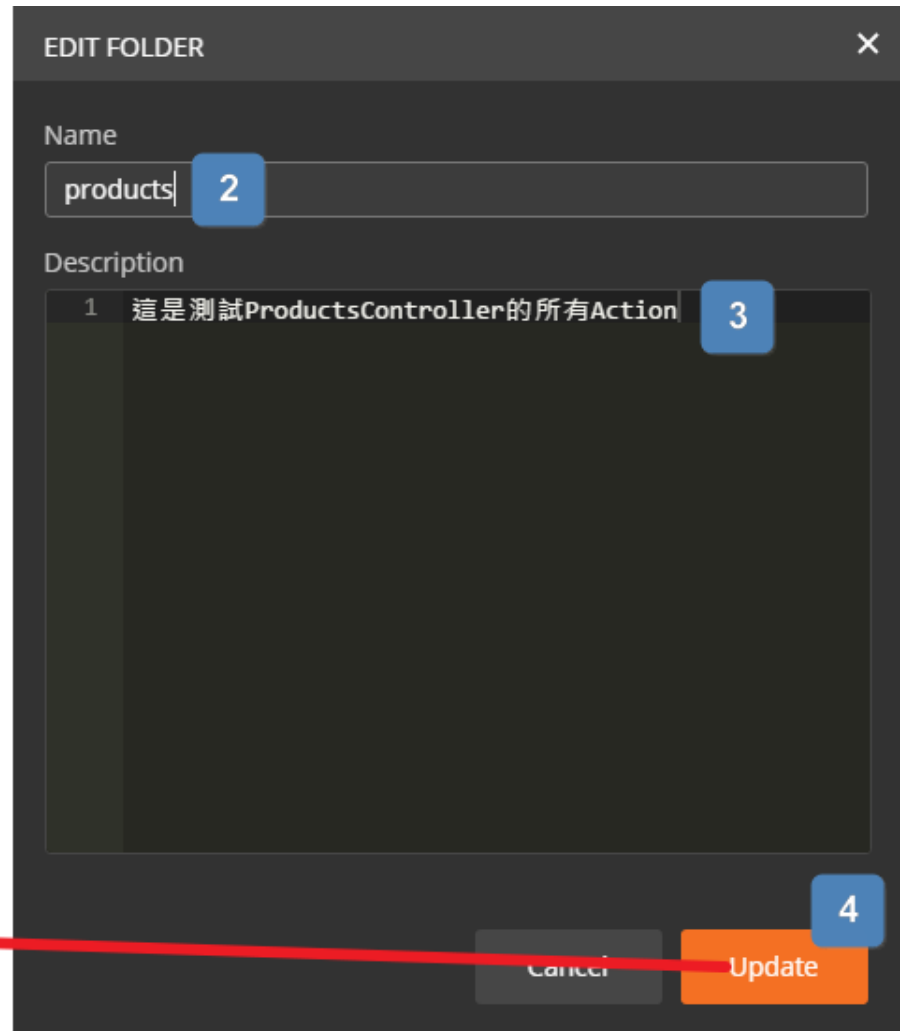
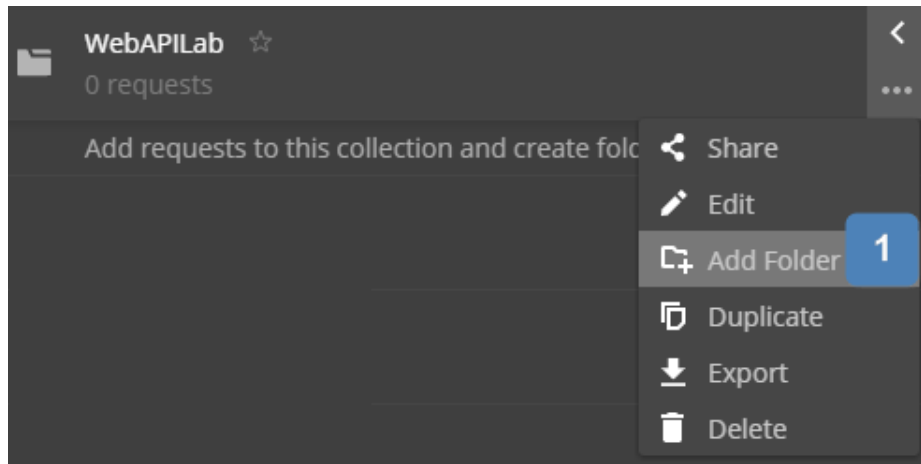


# 建立資料夾

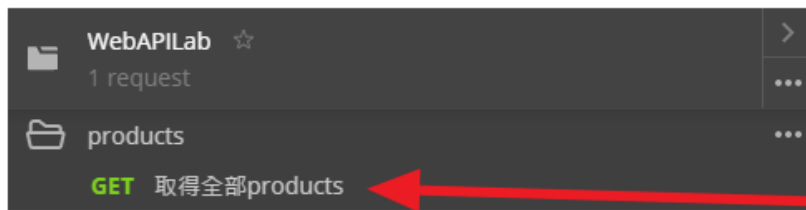
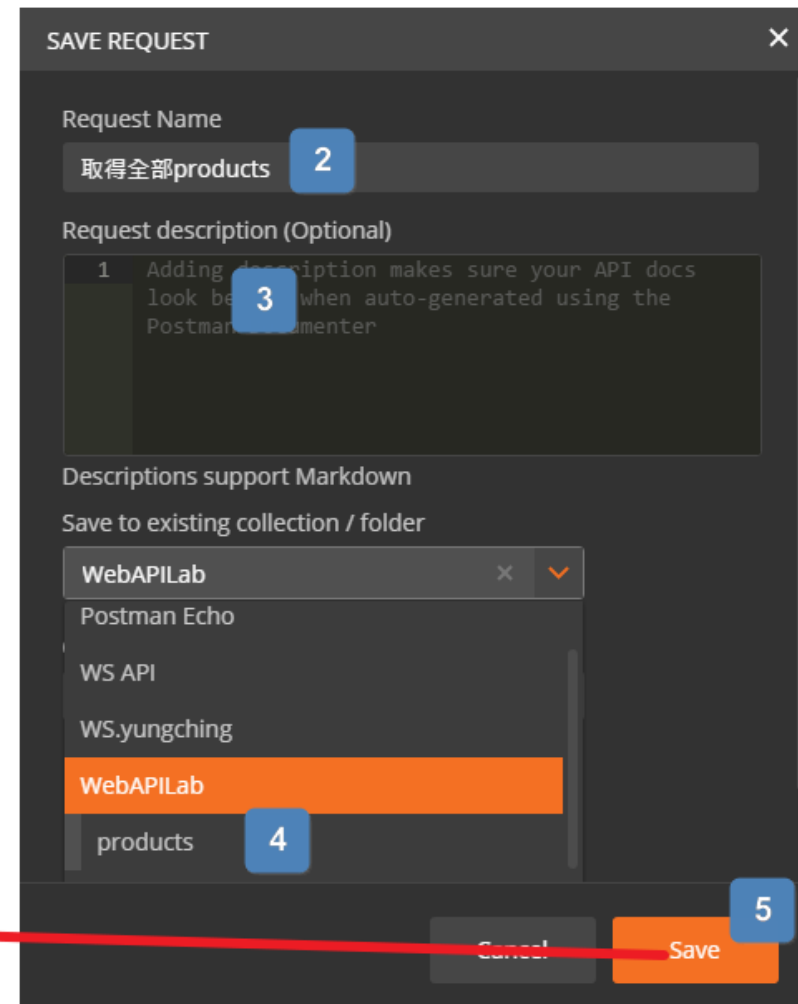
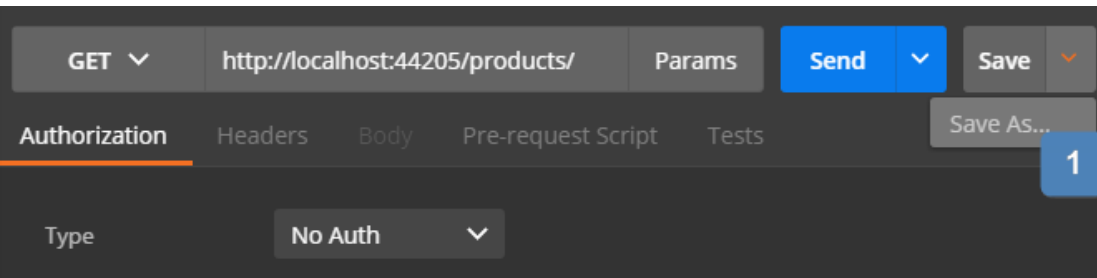




# 建立子資料夾



# Save



# Save

---

**GET** 取得全部products

**GET** 取得products用Name搜尋

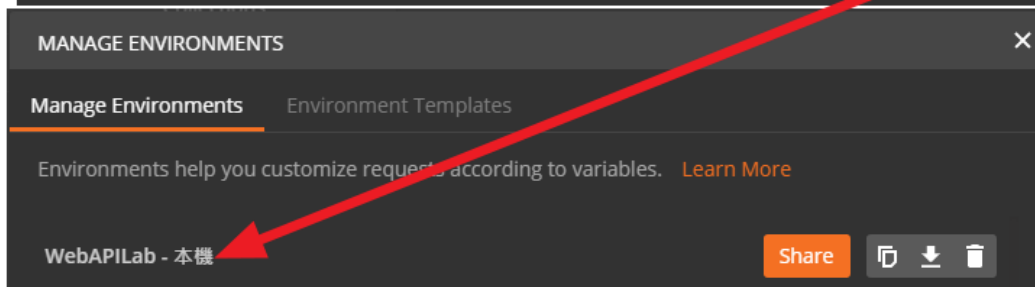
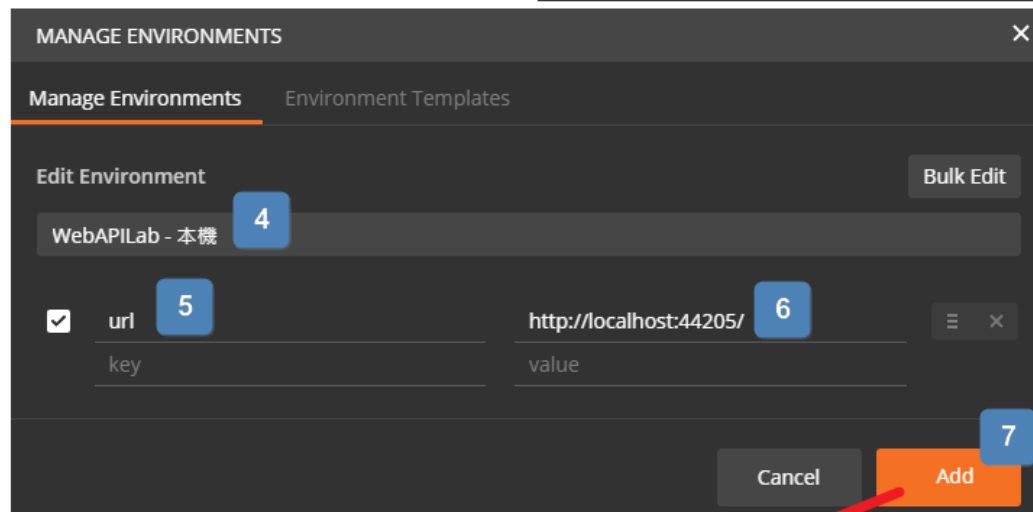
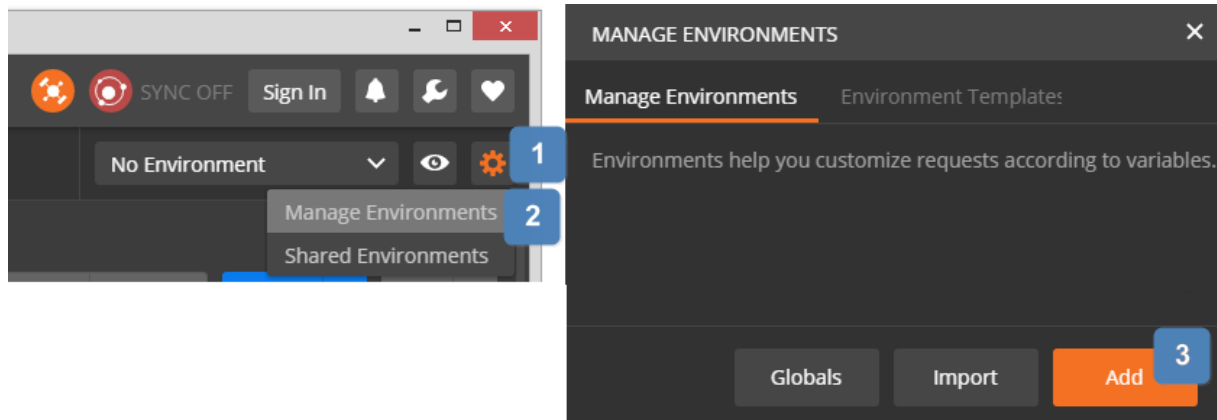
**GET** 取得products用指定Id

**PATCH** 修改 products name 用指定Id

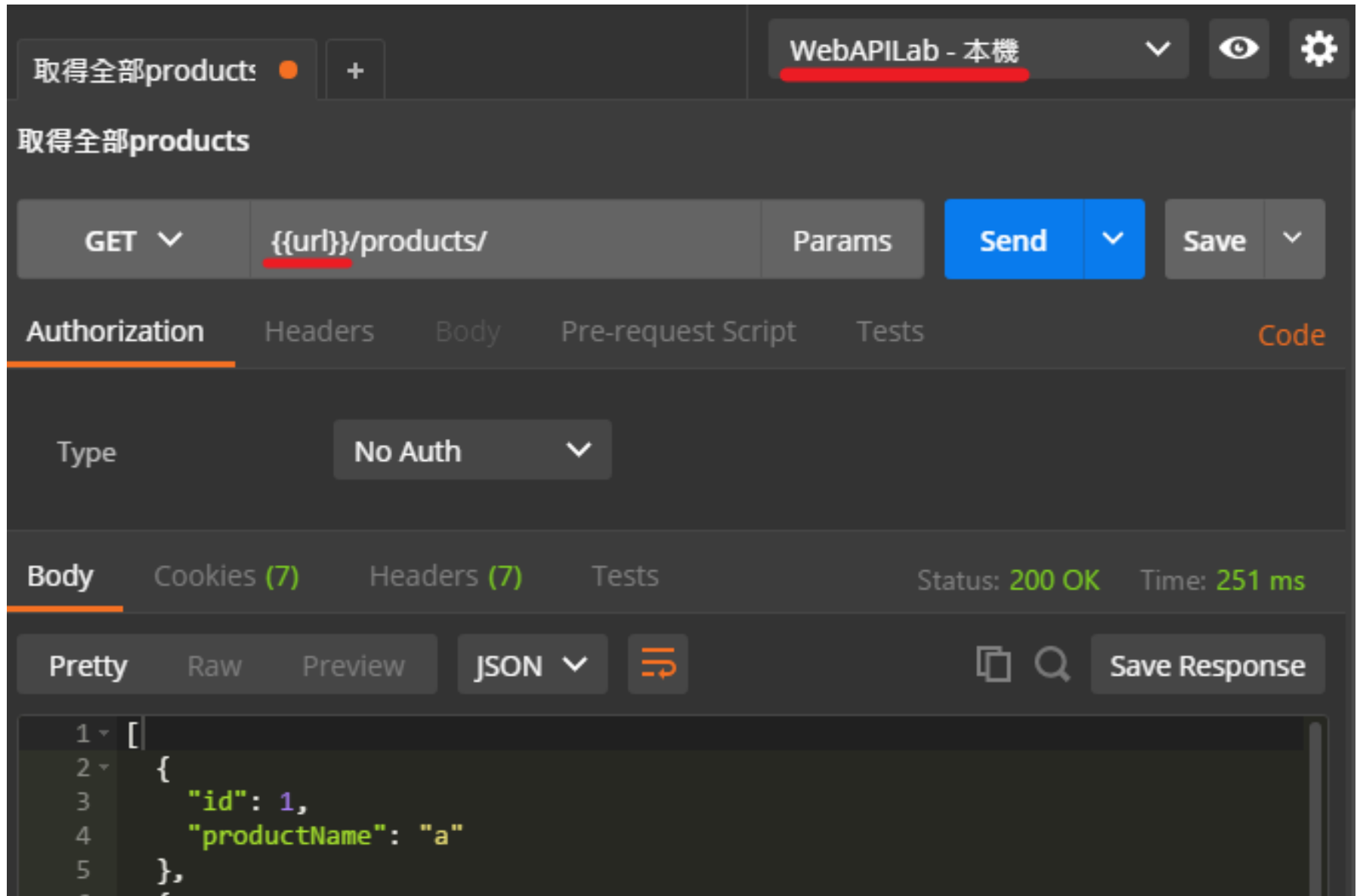
**DEL** 刪除 products name 用指定Id

**POST** 新增 products name

# 設定環境



# 設定環境



The screenshot displays the WebAPI Lab interface. At the top, a tab labeled "取得全部products" is active. Below it, the request method is set to "GET" and the URL is "{{url}}/products/". The "Params" tab is selected, and the "Send" button is highlighted in blue. The "Authorization" tab is also visible, showing "No Auth" selected. The "Body" tab is active, displaying a JSON response in "Pretty" format. The status bar indicates a successful response with "Status: 200 OK" and "Time: 251 ms".

取得全部products

WebAPILab - 本機

取得全部products

GET {{url}}/products/ Params Send Save

Authorization Headers Body Pre-request Script Tests Code

Type No Auth

Body Cookies (7) Headers (7) Tests Status: 200 OK Time: 251 ms

Pretty Raw Preview JSON Save Response

```
1 [
2   {
3     "id": 1,
4     "productName": "a"
5   },
6   ...
```

# 本機 測試網址 正式網址

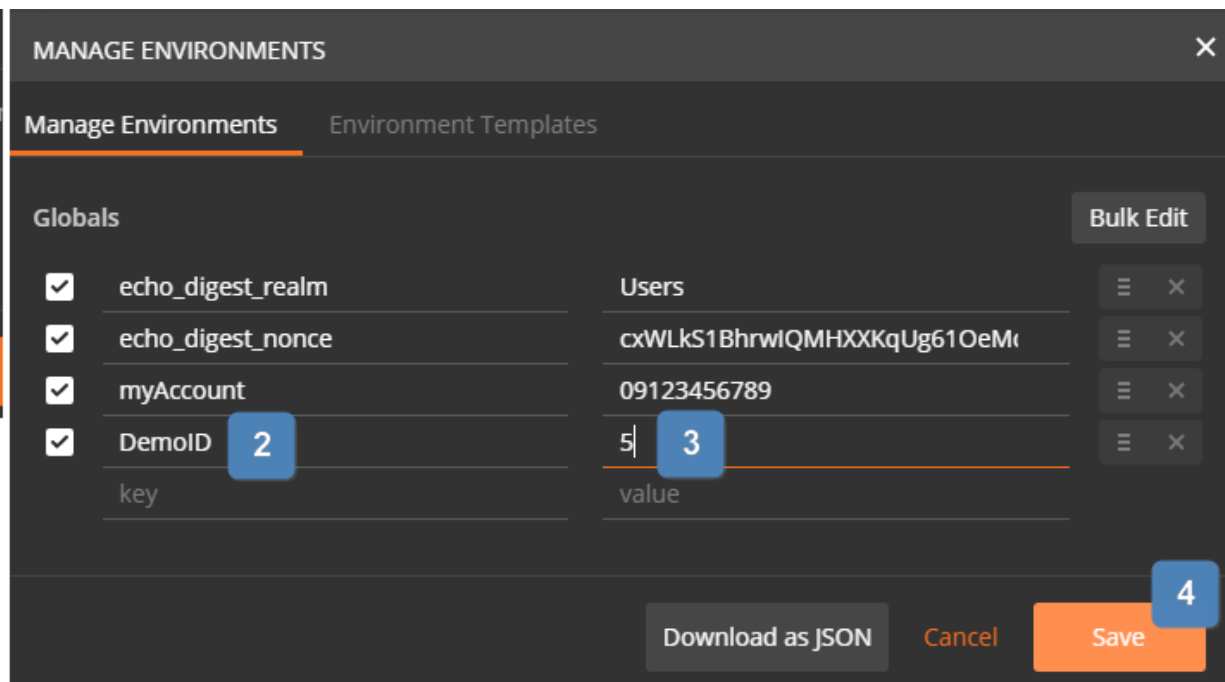
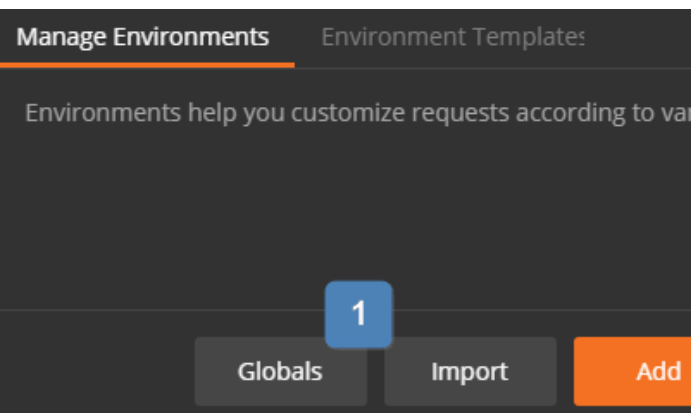


# 全域變數

---

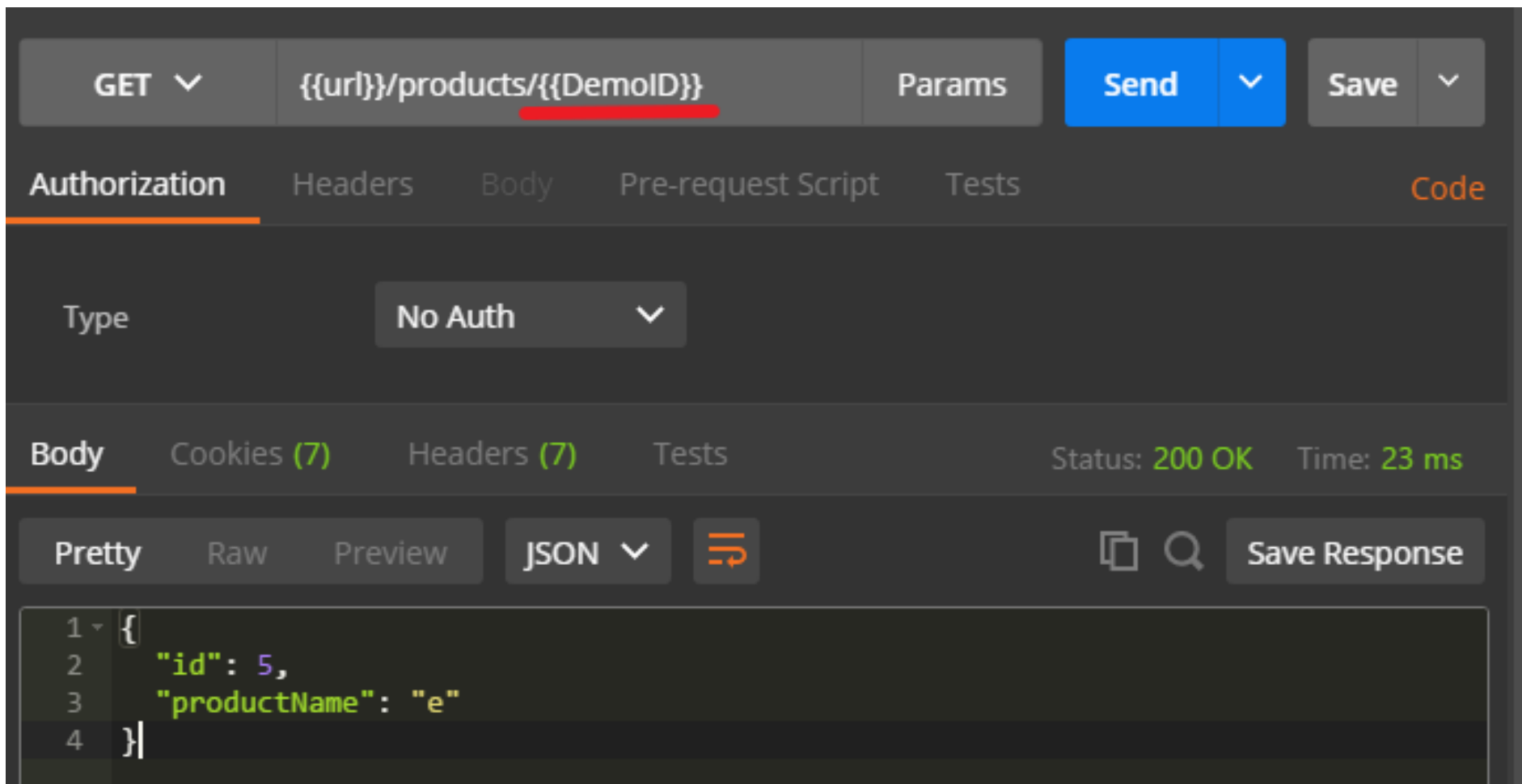
多個環境都要設定一樣的變數  
就可以設定到全域變數  
可能是帳號，密碼 ... 等

# 全域變數





# 全域變數

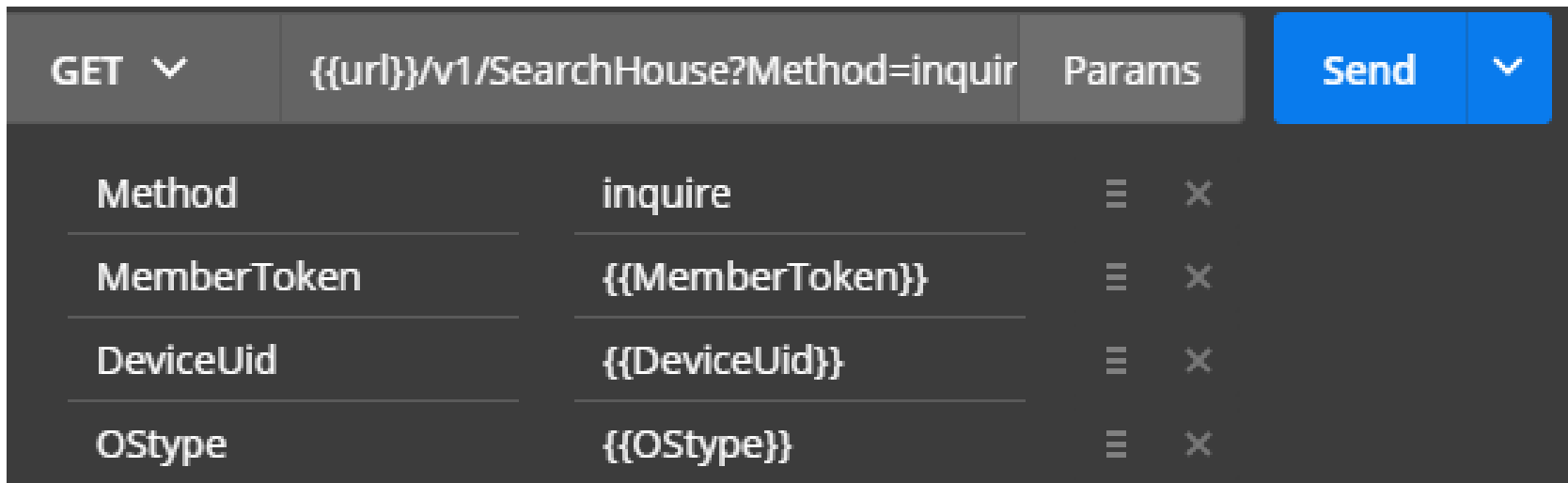
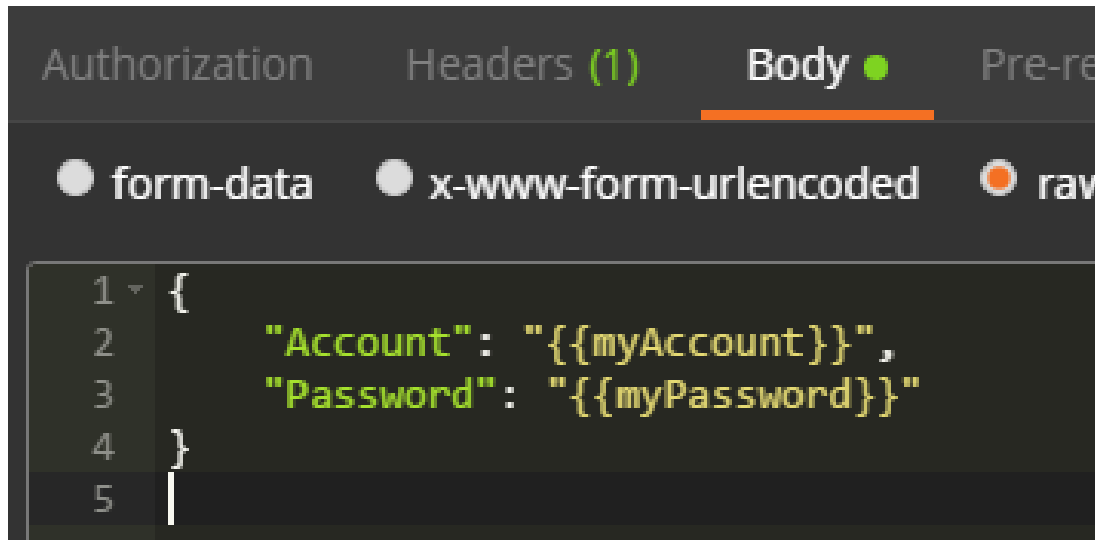


The screenshot displays a REST client interface with the following components:

- Request Bar:** Method `GET`, URL `{{url}}/products/{{DemoID}}` (with `DemoID` underlined in red), Params, `Send` button, and `Save` button.
- Request Tabs:** `Authorization` (active), `Headers`, `Body`, `Pre-request Script`, `Tests`, and `Code`.
- Auth Section:** Type `No Auth`.
- Response Section:** `Body` (active), `Cookies (7)`, `Headers (7)`, `Tests`, `Status: 200 OK`, and `Time: 23 ms`.
- Response Format:** `Pretty` (active), `Raw`, `Preview`, `JSON` (selected), and a `Save Response` button.
- Response Body:** A JSON object:

```
1 {  
2   "id": 5,  
3   "productName": "e"  
4 }
```

# 全域變數



謝謝大家