石頭閒語
Rock Saying

遊手好閒的石頭成的部落格
The Source Awakens! Make the source to be with you.

11月
5
2009

分類: Programming
標籤: 雲端運算 cloud computing web service REST RESTful
最近更新: 2009-11-05

RESTful 介面實作示範

大家好，我又來談 REST 了。雖然我早已在過去的實務工作中採用 RESTful 概念，但似乎在國內大多數人眼中， RESTful 還是個陌生的內容。因為我新任職的公司，才剛在專案中採用 RESTful 工作，還要大家去 survey 一下。也就在這過程中，我才發覺大伙兒對 RESTful 的認知還有不少偏差。最主要的一點還是把過去的 REST 作法混進來了。忽略了 RESTful 字尾的 **-ful** 所代表的意義。

雖然我一年多前在 REST and RESTful web service 就提過 RESTful 的內容了，不過當時主要放在 REST 和 RESTful 的差異上，假設讀者已經很熟悉 Web 架構與設計模式了。在當時的文章最後，附上的範例程式也僅僅是示範如何把舊的 REST 程式重構支援 RESTful 。沒有示範 RESTful 到底如何與客戶端互動。這篇文章就是在填這個坑。

REST 基礎觀念

REST (REpresentational State Transfer) 的概念來自於 Roy Thomas Fielding 寫的一篇文章《Architectural Styles and the Design of Network-based Software Architectures》。 其概念結合了 HTTP 與 URL 兩種協定，以及如何運用於網路軟體架構設計。

容我偷懶直接引用我之前的文章，用一段話來說明 RESTful: 以 URL 定位資源，根據 HTTP 內容指示操作動作與回應訊息。一個符合上述實作方式的網路服務，就稱之為 RESTful web service 。 有些文章則更進一步，將 ATOM 協定也加了進來，主要是看上 ATOM 格式的特點，將之運用於資源內容的更新工作。 有些 RESTful 文章還會強調要透過 HTTP Authorization 限制使用者存取資源的權限，而不是用表單加 Cookie。

規範書與文章參考 —

- RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1
- Wiki:List of HTTP headers, 內容較 RFC2616 新。
- Wiki:List of HTTP status codes, 內容較 RFC2616 新。
- Architectural Styles and the Design of Network-based Software Architectures
- XMLHttpRequest
- REST and RESTful web service

業界實務

千萬別以為 RESTful 只是輕量級、中小企業在用的次級品。事實上，眾多 Web 實務案例證明它才是 Web 架構的主流派，所謂 SOAP 不過是多此一舉。

目前的大型雲端運算提供者，如 Amazon Web Services 、Google、IBM 、Microsoft、Sun、GoGrid 等，都支援 RESTful 服務，甚至只提供 RESTful API。

業界的 **REST-like** 實作

我在 REST and RESTful web service 中，將傳統的 REST 作法稱為 REST-like ，這基本上不是我獨創的用法，例如知名的網路服務 Facebook 、 GoGrid 等，就將他們的 API 稱為 "REST-like"。

> The API uses a REST-like interface. This means that our Facebook method calls are made over the internet by sending HTTP GET or POST requests to the Facebook API REST server . Nearly any computer language can be used to communicate over HTTP with the REST server.
>
> — *API - Facebook Developer Wiki*

GoGrid 在 API 見此 [The GoGrid API]，有興趣了解 REST-like 的請自行參考。

附帶一提，PHP 主要開發商 Zend ，日前邀集了 IBM、Microsoft、GoGrid 一起推出了 Simple Cloud 計劃，它們將針對現行的主要雲端服務項目，以 PHP 實作一組 API 支援它們。對 PHP 開發者來說，這顯然是個好消息，也更進一步地支撐了 PHP 的企業級軟體開發能力。PHP 不是小企業才用的玩具，而是足以支撐大型企業專案的開發工具。

業界的 **RESTful** 實作

Sun Microsystem 公開了一套存取它們雲端運算服務的 Web service ，稱為 Sun Cloud API / API for Sun Cloud 。 這套 API 正是 RESTful 介面的。我取其中一個 method 來看看企業級的 RESTful 實作介面的外觀。

```
Get Cloud

Retrieve information about accessible resources.

Synopsis: GET {Well Known URI of the Service}
Request Headers: Accept, Authorization, X-Cloud-Client-Specification-Version.
Request Message Body: N/A.
Response Headers: Content-Length, Content-Type.
Response Message Body: Cloud.
Response Status: 200, 400, 401, 403, 404.
```

Cloud API Specification - Requests to Cloud Resources

Synopsis 指出這是 GET 方法的說明。在 Request Headers 中的 Authorization 表示 Sun Cloud API 是採用 HTTP Authorization 驗證方式，而不是用網頁表單login。 方法的狀態回傳值也是透過 HTTP Status code ，此例中表明此資源可能傳回 5種狀態， 200 正常，404 是找不到指定內容，401 是未認證，403 是未授權、400 是錯誤請求。 初涉 RESTful 的人，往往未了解 Status code 的重要性， Status code 會影響 proxy 對資源內容的快取，也會影響 client 的處理。

HTTP Authorization 內容請參考 RFC2616 14.8 節。Apache Http Server 實作內容請參考 Authentication, Authorization and Access Control。

Amazon Web Service 也支援 RESTful API ，它們的 API 文件雖然不像 Sun Cloud API 那般清楚條列參數與回傳狀態，但卻清楚列出使用服務過程中，透過 HTTP 交涉的訊息內容。 對於熟悉 HTTP 協定的開發者，是不錯的示範。各位可以看看 [Amazon S3 GET Service] 的內容，連 HTTP header 都列出了。

Demo

請各位將下列的兩個 PHP 程式碼複製儲存到自己的主機上，放在你的網頁空間下。 當然啦你的網頁服務要能跑 PHP ，這麼基本的事就不多說了。

index.php —

```php
<?php
class Control {
    static function exceptionResponse($statusCode, $message) {
        header("HTTP/1.0 {$statusCode} {$message}");
        echo "{$statusCode} {$message}";
        exit;
    }

    function index() {
        echo 'index...';
    }
}

interface RESTfulInterface {
    public function restGet($segments);
    public function restPut($segments);
    public function restPost($segments);
    public function restDelete($segments);
}

class Container extends Control {
    private $control = false;
    private $segments = false;

    function __construct() {
        if ( !isset($_SERVER['PATH_INFO']) or $_SERVER['PATH_INFO'] == '/') {
            // $this->control = $this->segments = false;
            return;
        }

        $this->segments = explode('/', $_SERVER['PATH_INFO']);
        array_shift($this->segments); // first element always is an empty string.

        $controlName = ucfirst(array_shift($this->segments));

        if ( !class_exists($controlName) ) {
            $controlFilepath = $controlName . '.php';

            if ( file_exists($controlFilepath) ) { // 載入客戶要求的 control
                require_once $controlFilepath;
            }
            else { // 找不到客戶要求的 control
                self::exceptionResponse(503, 'Service Unavailable!');
                // 回傳 501 (Not Implemented) 或 503.
                // See also: RFC 2616
            }
        }
```

```
        $this->control = new $controlName;
    }

    function index() {
        echo 'index.php/{control name}/{object id}';
    }

    function run() {
        if ( $this->control === false) {
            return $this->index();
        }

        if ( empty($this->segments) ) { // Without parameter
            return $this->control->index();
        }

        //request resource by RESTful way.
        //$method = $this->restMethodname;
        $method = 'rest' . ucfirst(strtolower($_SERVER['REQUEST_METHOD']));
        if ( !method_exists($this->control, $method) ) {
            self::exceptionResponse(405, 'Method not Allowed!');
        }

        $arguments = $this->segments;

        $this->control->$method($arguments);
    }
} //end class Container

$container = new Container();

$container->run();

?>
```

Mock.php —

```
<?php
class Mock extends Control implements RESTfulInterface {
    function restPost($segments) {
        echo 'Create resource';
        echo '<br/><pre>';
        print_r($_POST);
        echo '</pre>';
    }

    function restGet($segments) {
        $id = $segments[0];
        if ($id == 'rock')
            echo 'Read resource: ' . $segments[0]; // id
        else
            self::exceptionResponse(404, 'Not found');
    }

    function restPut($segments) {
        echo 'Update resource: ' . $segments[0];
        echo '<br/> you put data: ' . file_get_contents('php://input'); // read the raw put data.
    }

    function restDelete($segments) {
        echo 'Delete resource: ' . $segments[0];
    }
}
?>
```

眼尖的 PHP 程序員應該會注意到，我雖然定義了 `RESTfulInterface` ，但其實在程式碼中根本沒有任何作用。 因為 PHP 的動態能力與反射能力根本不需要仰賴介面來檢查方法。 我定義 `RESTfulInterface` 只是在唬弄 Java 語言的使用者。

關於 PHP 的反射能力，我之前談過不少，例如PHP5 的動態函數/行為調用效率測試 、Reflection 於設計 Framework 時之安全性作用 。要從反射能力上挑 Java 的痛腳實在太容易，派 PHP 去叫陣就夠了。

當各位將上述的 PHP 程式儲放在可用的網頁空間之後，可自行在瀏覽器中輸入下面的案例網址，觀看反應。 別忘了將網址換成你實際的位址...

- Case 1: Container 顯示基本用法.
  http://your_host/restful_demo/index.php
- Case 2: 要求的服務不存在。
  http://your_host/restful_demo/index.php/nothing

  回應 503

- Case 3: 要求已存在的服務 mock ，但未加參數。
  http://your_host/restful_demo/index.php/mock

  RESTful 要求透過 HTTP status code 回傳執行的結果代碼。 在此處找不到客戶要求的 control ，按照這個狀態，應回傳 501 (Not Implemented) 或 503 (Service Unavailable)。(*1) 實務慣例則可以導引到服務說明頁面，說明服務的功能與用法。

- Case 4: 向服務 mock 讀取 xyz 的內容 （此目標不存在）
  http://192.168.1.33/workspace/restful_demo/index.php/mock/xyz

  目標資源不存在，回應 404 Not Found。初學者在實作 RESTful 服務時最常犯錯的地方，就是在這種情況下回傳 status code 200。你的服務明明就找不到使用者要查的資源，為何回應 200，告訴使用者、瀏覽器、proxy 等人找到資料了？要仔細理解 HTTP 作為一個應用程式協定時，各項 status code 所代表的意義。

- Case 5: 向服務 mock 讀取 rock 的內容
  http://your_host/restful_demo/index.php/mock/rock

*1 有些瀏覽器在收到501或503的回應時，會直接導向它自己的訊息畫面，這時使用者就看不到我們放在回應內容本文中的錯誤訊息。

HTML Form 僅規範了 GET 和 POST 兩種方法，若要在瀏覽器端向 Web 服務發送 PUT 和 DELETE 方法，則需要透過 JavaScript 操作 XMLHttpRequest object 才可 實現。

所以接下來的示範內容，會用 JavaScript 示範。請將下列的 demo.html 儲放在與上述 PHP 程式相同的目錄下。

demo.html —

```
<html>
<script src="http://www.google.com/jsapi"></script>
<script>

var mockUrl, options, canvas;

google.load("jquery", "1.3.2");
google.setOnLoadCallback(function(){
    //Enable Ajax object.

    mockUrl = 'http://localhost/test/restful_demo/index.php/mock/';
    options = {
        url: mockUrl + 'rock',
        async: false,
        type: 'get',
```

```
        data: ''
    };

    canvas = $('#canvas');

    alert('Case 6: 新增資源 (POST)\n' + demoCase6);
    demoCase6();

    alert('Next Case 7: 更新資源 (PUT)\n' + demoCase7);
    demoCase7();

    alert('Next Case 8: 刪除資源 (DELETE)\n' + demoCase8);
    demoCase8();

    alert('Next Case 9: 找不到指定資源\n' + demoCase9);
    demoCase9();

    alert('End.');
});

function
demoCase6() { //Case 6: 新增資源 (POST)

    options.type = 'post';
    options.data = 'abc=123&def=567';
    canvas.html( $.ajax(options).responseText );
}

function
demoCase7() { //Case 7: 更新資源 (PUT)

    options.type = 'put';
    options.data = 'abc=764';
    canvas.html( $.ajax(options).responseText );
}

function
demoCase8() { //Case 8: 刪除資源 (DELETE)

    options.type = 'delete';
    canvas.html( $.ajax(options).responseText );
}

function
demoCase9() { //Case 9: 找不到指定資源

    options.url = mockUrl + 'xyz';
    options.type = 'get';
    options.complete = function(xhr, statusText) {
        alert('Status code: ' + xhr.status);
    };
    $.ajax(options);
}

</script>

<div id="canvas" style="width:500;height:300;border:2px solid black">
</div>

</html>
```

- Case 6: 新增資源 (POST)

```
options.type = 'post';
options.data = 'abc=123&def=567';
canvas.html( $.ajax(options).responseText );
```

- Case 7: 更新資源 (PUT)

```
options.type = 'put';
options.data = 'abc=764';
canvas.html( $.ajax(options).responseText );
```

- Case 8: 刪除資源 (DELETE)

```
options.type = 'delete';
canvas.html( $.ajax(options).responseText );
```

- Case 9: 找不到指定資源

```
options.url = mockUrl + 'xyz';
options.type = 'get';
options.complete = function(xhr, statusText) {
    alert('Status code: ' + xhr.status);
};
$.ajax(options);
```

RESTful 示範到此結束，Enjoy RESTful.

相關文章 —

- index.php - CommonGateway 介紹

> 樂多舊網址: http://blog.roodo.com/rocksaying/archives/10568163.html

---

樂多舊回應 —

> leebig1982@gmail.com(Lucas) (#comment-20160965)
> Wed, 09 Dec 2009 11:20:37 +0800
>
> 感謝石頭大的分享，小弟本身有一點很疑惑:REST是否適合使用在撰寫API方面，比如有關金流/WEB ATM/線上刷卡這類的用途。煩請石頭大略為指點，感激不盡

遊手好閒的石頭成
@tw_rocksaying
Host on GitHub