RESTful API 設計慣例

API 版本分析與使用時機

為何使用 web framework

Gin MVC 規格

Restful API 設計慣例

若要增加 header, 請記得在 CORS 中增加對應的 `Access-Control-Allow-Headers`

See Cors Access-Control-Allow-Headers wildcard being ignored?

API 版本

使用 `URI` 代表版本:

1. 沒有版號預設為 `v1` 版, 例: `http://<host>/api/v1/<resource...>`

其它 versioning 方式

- 使用 domain name - `<version>.service`，適合用在另一版用另一程式語言實作
- Header - 讓 URI 維持不動
- Query Parameter - 讓 URI 維持不動

使用何種機制來指定 API 的 Version 是哲學問題

頁碼分頁

當 API 要提供分頁時，一律以下列規格設計參數

在文件使用 **paging-number**，代表此 API 有分頁功能

在 HTTP Header 中，自定義下列 Header:

Request

- `page-size` - 每頁筆數，API 一定要提供預設值(值過大會以可允最大值覆寫)
- `page-pos` - 第 N 頁，從 1(預設值) 開始
- `order-by` - `<prop_1>#<dir>:<prop_2>#<dir>` - 以 `:` 分隔的多重欄位排序語法
  - `<prop_1>` - 屬性名稱
  - `<dir>` - 方向
  - 範例: `name#asc:connection_id:status#desc` - 先以 **name** 排序(遞增)，再以 **connection_id**(預設方向)，最後以 **status** 排序(遞減)

Response

- `page-size` - 每頁筆數，Server 回應的安全值
- `page-pos` - Server 回應的安全值
- `total-count` - 總筆數，API 可不提供此 header
- `page-more` - `true`, `false` 當總筆數會影響效能時，此 header 代表是否有下一頁

使用 Header 的好處是，輸出多筆資料的 API 不需使用 JSON Object 的格式

指標分頁

在文件使用 **paging-pointer**，代表此 API 有分頁功能

指標應包含排序資訊等分頁額外資料

在 HTTP Header 中，自定義下列 Header:

Request

- `page-ptr: <location>` - 可以是 URL, HASH code 或 URI，代表開始的指標

Response

- `previous-page: <location>` - 可以是 URL, HASH code 或 URI，代表前一頁的指標
- `next-page: <location>` - 可以是 URL, HASH code 或 URI，代表下一頁的指標

複雜分頁

由 API 的 JSON 內容自行定義

Parameters

關於參數，或資料欄位的屬性

- `Mandatory` - Client 一定要提供的資料
- *[auto-generated]* - 系統自動產生的資料，例如 internal key，**不能被新增或修改**
- *[new]* - 只能被新增，不能被修改
- *[null]* - Server 可能傳回 null 值
- *[missed]* - 這個屬性可能完全不顯示出來
- *[empty]* - JSON 的 array 可能是空陣列 `"hosts" : []`
- *[unix time]* - 值為 UNIX Time, 例: `1471995721` (2016-08-23T23:42:01Z)

全域錯誤

當 API 有下列錯誤時，一律以下列規格產生結果

Response `400`

(`application/json`) 資料驗證錯誤，例如密碼太短或必需要填入非空白字串通常是 Client Side 就能檢查的錯誤

```
{
  "http_status": 400,
  "error_code" : -1,
  "error_content" : [
    { /* 屬性的名稱與錯誤原因 */ }
  ]
}
```

Response `409`

(`application/json`) 唯一性錯誤，必需由 Server Side 檢查

```
{
  "http_status": 409,
  "error_code" : -1,
  "error_content" : [
    { /* 屬性的名稱與錯誤原因 */ }
  ]
}
```

Response `500`

(`application/json`) 非預期的錯誤，例資料庫無法連到或 golang panic

```
{
  "http_status": 500,
```

```
  "error_code" : -1,
  "error_message": "dial tcp 192.168.20.50:3306: connectex: No connection
...", // 錯誤訊息
  "error_stack": [
      "github.com/Proj/a.go:458",
      "github.com/Proj/a.go:332"
  ]
}
```

Response `404`

(`application/json`) 資源不存在

```
{
  "http_status": 404,
  "error_code" : -1,
  "uri": "/no-such-resource"
}
```

Response `405`

(`application/json`) 不允許使用該方法(method)

```
{
  "http_status": 405,
  "error_code" : -1,
  "method": "POST",
  "uri": "/no-such-resource"
}
```

Response `401`

(`application/json`) 沒有登入的錯誤，可能用 `404` 來提高安全度

```
{
  "http_status": 401,
  "error_code" : -1
}
```

Response `403`

(`application/json`) 有登入(系統知道你是誰)，但沒有權限存取資源可能用 `404` 來提高安全度

```
{
  "http_status": 403,
  "error_code" : -1
}
```

錯誤分層

第一層: HTTP Status

第二層: Response Body

例：下列例子中，error_code 為 `1` 有兩種，分別以 `400` 與 `401` 區分

Status	JSON Body	Comment
400	`{ "error_code" : 1 }`	`400` AND `1`
401	`{ "error_code" : 1 }`	`401` AND `1`
401	`{ "error_code" : 2 }`	`401` AND `2`

Last modified on 2017-02-15T10:06:58+08:00