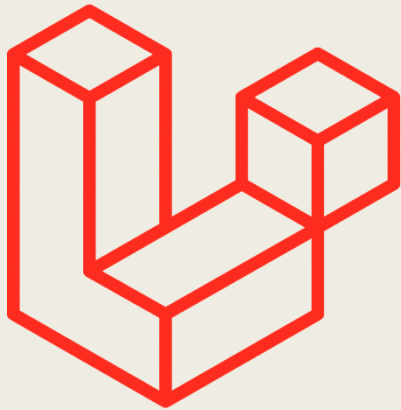


# RESTful API



# 定義

- 遵守 REST 架構所設計的 Web API 稱為 RESTful API
- 網址不會使用帶參數的格式，例如 `http://.../query.php?userid=A01`，會使用 `http://.../query/A01`
- 回傳的資料格式通常為 JSON、XML 或 HTML 網頁
- 向 Web server 要求資源時，不止 GET 與 POST 這兩種方法，但表單只支援 GET 與 POST

# 方法種類

- GET
  - 取得資料
- POST
  - 新增資料
- PUT
  - 完整更新資料或新增
- DELETE
  - 刪除資料
- PATCH
  - 更新部分資料
- HEAD
  - 與 POST 同，只是不傳回資料內容，僅傳回表頭資料而已
  - **Laravel 不支援**
- OPTIONS
  - 傳回該資源的資訊，例如可以使用哪些方法

# 建立 Web API

- 開啟 Laravel 路由的 api.php，語法與 web.php 一樣，只不過沒有 view
- 網址加上 api 即可，例如
  - `http://.../laravel/public/api/...`

# Laravel 的表單

- 防止跨域攻擊 ( 看到 419 Page Expired 錯誤時 )

```
<form ...>  
    @csrf  
</form>
```

- 額外設定 PUT、DELETE...隱藏欄位

```
<form method="post">  
    @method('PUT')  
</form>
```

# FETCH

# 基本用法

- AJAX 非同步 JS 技術，可減少網路頻寬消耗與大幅提升使用者體驗
- 文件
  - [https://developer.mozilla.org/en-US/docs/Web/API/Fetch\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API)
- 基本用法：

```
<script>
window.onload = function() {
  document.getElementById("bn").onclick = function() {
    fetch("news.txt")
      .then((response) => {
        return response.text()
      })
      .then((text) => {
        document.getElementById("test").innerHTML =
text
      })
  }
}
</script>
<body>
<button id="bn">click</button><p></p>
<div id="test"></div>
</body>
```

# 取得整個 body 內容

## ■ 後端

Laravel

```
$body = $request->all()
```

```
$body = file_get_contents('php://input');
```

## ■ 前端

標準 PHP 函數

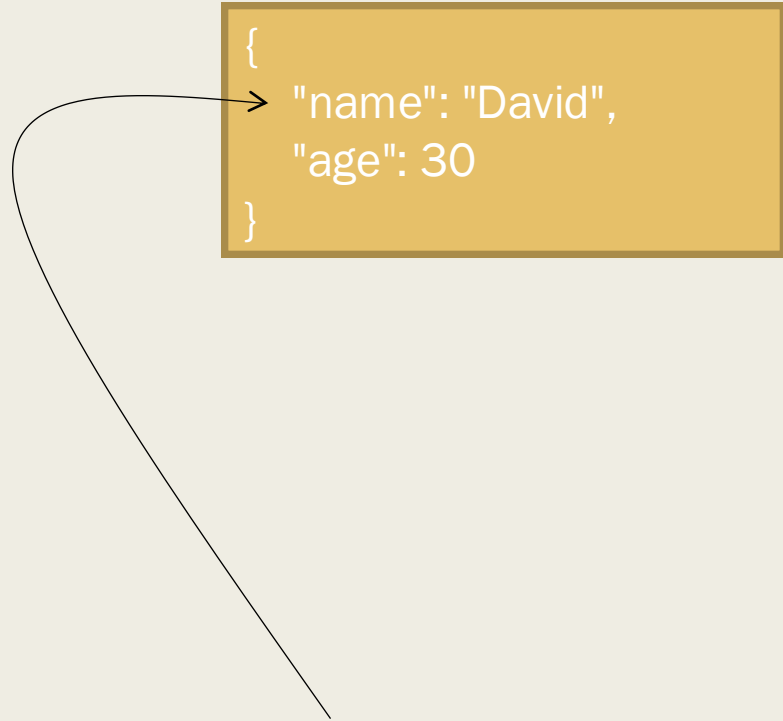
```
fetch('/url', {  
  method: 'post',  
  body: '{"name": "Betty"}',  
  headers: {  
    'Accept': 'application/json',  
    'Content-Type': 'application/json'  
  }  
})
```



# 回傳型態為 json

- 預設值為 get
- 使用 json() 解析傳回的 json 字串

```
fetch("demo.json", {method: "get"})  
  .then((response) => {  
    return response.json()  
  })  
  .then((jsonObj) => {  
    document.getElementById("test").innerHTML = jsonObj.name  
  })
```



A yellow box containing a JSON object: { "name": "David", "age": 30 }. An arrow points from the 'name' property value to the 'name' property access in the code snippet below.

```
{  
  "name": "David",  
  "age": 30  
}
```

# 回傳型態為 BLOB

- 圖片、聲音、影像、zip...等

```
fetch('URL')  
  .then(function(response) {  
    return response.blob()  
  })  
  .then(function(blob) {  
    const url = URL.createObjectURL(blob)  
    document.getElementById('image').src = url  
  })
```

# 與表單結合

- 注意 submit 用的按鈕不要放在表單內，否則 submit 後會實際更新網頁
- 另一種作法見下一張投影片

```
<body>
<form id="myform">
  <input type="file" name="file"><p>
</form>
<button id="bn">Click</button>
<hr>
<div id="result"></div>
</body>
```

```
window.onload = function() {
  document.getElementById("bn").onclick = function() {
    var formData = new FormData(document.getElementById("myform"))
    fetch("uploadfile.php", {
      method: "post",
      body: formData
    })
    .then((response) => {
      return response.text()
    })
    .then((data) => {
      document.getElementById("result").innerHTML = data
    })
    .catch((error) => {
      console.error('Error:', error)
    })
  }
}
</script>
```

# 若 submit 按鈕放在表單內

- 此時一定要呼叫 `preventDefault()` 停掉預設的 submit 事件

```
document.getElementById("bn").onclick = function(event) {  
    event.preventDefault()  
    ...  
}
```

```
<form id="myform">  
  <input type="file" name="file"><p></p>  
  <button id="bn">Click</button>  
</form>
```

跨域存取

# 同源政策

- 瀏覽器因為安全因素，不允許網頁上的資料來自於不同的網站，這稱為同源政策
- 瀏覽器只允許少數標籤可以跨來源，例如 `<img>` `<script>`
- 允許跨源必須是後端同意，意思是 PHP 要送出允許跨源的表頭資料
  - 例如：`Access-Control-Allow-Origin: *`

# Laravel 的跨源處理

- 在 API 中的路由，已經自動允許跨源
- 在 Web 中的路由，必須手動允許跨源

```
Route::get('/test', function() {  
    return response('hello', 200)  
        ->header('Access-Control-Allow-Origin', '*');  
});
```