B0929045 叶子晴 資工二甲

跨來源資源共用(Cross-Origin Resource Sharing) ，則CORS，是一種使用額外 HTTP 標頭令目前瀏覽網站的使用者代理能取得存取其他來源伺服器特定資源權限的機制。當使用者代理請求一個不是目前文件來源——例如來自於不同網域（Domain）、通訊協定（Protocol）或通訊埠（Port）的資源時，會建立一個跨來源 HTTP 請求（Cross-Origin HTTP Request）來獲取資訊。基於安全性考量，程式碼所發出的跨來源 HTTP 請求會受到限制，代表網路應用程式所使用的 API 除非使用 CORS 標頭，否則只能請求與應用程式相同網域的 HTTP 資源。其優點在於在提供了網頁伺服器跨網域的存取控制，同時增加跨網域資料傳輸的安全性。常用在於是利用 XML Http Request 或 Fetch API 進行跨站請求、跨網域 CSS 的 font-face 網頁字體(CSS 樣式表)或是指令碼。

其標準的運作方式是藉由加入新的 [HTTP 標頭](https://developer.mozilla.org/zh-TW/docs/Web/HTTP/Headers)讓伺服器能夠描述來源資訊以提供予瀏覽器讀取。

CORS 分為簡單請求（Simple requests）和預檢請求（Preflighted requests）兩種，發出 CORS Request 時瀏覽器會自動在 Request Header 加上目前的 Origin。

在簡單請求（Simple Requests）中，如果 Request Method 是GET、HEAD、POST 其一，且 Request Header 的 Content-Type 是application/x-www-form-urlencoded、multipart/form-data或text/plain其中一種的話那就是簡單請求，後端則不需再做額外設定。

預檢請求（Preflighted Requests）中，就是為不符合簡單請求的規則作檢查，例如使用了 PUT、DELETE 等 Method 或者 Content-Type 是 application/json，在送出該 Request 之前，瀏覽器會先進行一次預檢（Preflight），發送預檢請求時瀏覽器會先以 OPTIONS Method 問候一下後端，後端收到 Request 後，只要 Request Header 的 Origin 和 Access-Control-Request- 系列都在 Response Header 的列表中，瀏覽器就會發出正式的 Request，和簡單請求不同的是如果沒有通過預檢，就不會發送 Request。

另外，還有一種是可附帶身分驗證的請求，XML Http Request 或 Fetch 在 CORS 中有趣的功能為傳送基於 HTTP Cookies 和 HTTP 認證（Authentication）資訊的「身分驗證（Credentialed）」請求。預設 CORS Request 都是匿名（Anonymous）發送，想要帶上 Cookies 或是收到 Cookies 需要在前後端都加入一點設定。前端以 Fetch 為例，無論是簡單還是預檢請求，加上一個設定值後瀏覽器在發出 CORS Request 時就會帶上 Cookies，同時 Response header 中的 Set-Cookie 才會生效。在簡單請求的狀況下，收到正確 Response 後一切都會正常運作；在預檢請求的情況下，只有正式 Response 中的 Set-Cookie 才會生效。

在回應一個身分驗證請求時，伺服器必須於 Access-Control-Allow-Origin 標頭值中指定一個來源，而不是使用「\*」萬用字元（wildcard）。例如實際的來源是來自一個網址(http://for.example) 其中包含了一個 Cookie 標頭，若 Access-Control-Allow-Origin 標頭為「\*」，請求將會失敗。舉個例子，當 Access-Control-Allow-Origin 標頭值為「http://for.example」，而不是「\*」萬用字元，所以身分驗證證明內容被回傳予呼叫的網站內容中。如果 Set-Cookie 回應標頭也設定了另一個 cookie時，當請求失敗，就會取決於所使用的 API而拋出一個錯誤。另外，若是使用第三方 Cookies的話，在 CORS 回應中設定的 Cookies 會受制於一般的第三方 Cookie 政策。如果使用者將其瀏覽器設定為拒絕所有第三方 Cookies，則 Cookies 不會被保存。