B0929036 林湘庭

Cross-Origin Resourse Sharing (CORS)

CORS 是一種瀏覽器跨網域連線的方式，他透過設定 HTTP 的 header，可以規範瀏覽器在進行跨網域連線時可以存取的資料權限與範圍，像是那些來源可以存取，或者哪些 HTTP verb, header 的 request 可以存取。

瀏覽器會根據送出的 request 的 HTTP verb 與 header，判斷這個 request 是一個簡單請求（simple request）或是非簡單請求。判斷是簡單請求還是非簡單請求的方式如下：

簡單請求（simple request）是不會觸發 CORS 預檢的，他是滿足以下所有條件的請求：

- 允許的方法之一：GET 、 HEAD 、 POST
- 除了由用戶代理自動設置的標頭（例如：Connection 或 User-Agent 在 Fetch 規範中定義為禁止標頭名稱的其他標頭）之外，唯一允許手動設置的標頭是 Fetch 規範定義為的標頭一個 CORS-safelisted request-header，它們是：Accept 、 Accept-Language 、 Content-Language 、 Content-Type（請注意下面的附加要求） 、 Range（僅使用簡單的範圍標頭值；例如，bytes=256-或 bytes=127-255）
- 標頭中指定的媒體類型允許的唯一類型/子類型組合 Content-Type 是：application/x-www-form-urlencoded 、 multipart/form-data 、 text/plain
- 如果使用 XMLHttpRequest 對象發出請求，則不會在請求中使用的屬性返回的對像上註冊任何事件偵聽器 XMLHttpRequest.upload；也就是說，給定一個 XMLHttpRequest 實例 xhr，沒有代碼調用 xhr.upload.addEventListener()添加事件監聽器來監控上傳。
- 請求中未使用任何 ReadableStream 對象。

如果是一個非簡單請求，就會進行 CORS 預檢(CORS preflight)，先對伺服器送出一個 verb 為 OPTION 的 preflight request，它會帶有特定的 header 告訴伺服器接下來的 request 需要哪些跨網域連線的權限，以確定實際請求是否可以安全發送。

如果滿足以下任一項條件時會發出預檢請求：

- 某些請求方法為以下其中之一：PUT（英文）、刪除 (en-US)、CONNECT、選項（英文）、TRACE（英文）、補丁（英文）

- 某些除了用戶代理自動設置的標頭（例如 Connection、User-Agent 或任何請求規範[獲取規範]中定義的“禁止使用的標頭名稱[禁止標頭名稱]”中的標頭）之外，包含了這些除了在任何請求規格（Fetch spec）中定義為“CORS 安全列表請求標頭（CORS-safelisted request-header）”以外的標頭，具體如下：Accept、接受語言（en-US）、內容語言（en-US）、Content-Type（但請注意下面的額外要求）、Last-Event-ID、DPR、Save-Data、Viewport-Width、Width
- Content-Type 市場上的一些標有除名以外的其他頭值 ：application/x-www-form-urlencoded、multipart/form-data、text/plain

當伺服器收到 preflight 後，就會回傳帶有特定 header 的 response 給瀏覽器，告訴它有哪些權限是允許的。

瀏覽器取得伺服器的 response 後，如果符合連線權限，就會送出真正的 request。如果發現權限不符，就會出現錯誤訊息而中斷送出 request 的步驟。

如果要做跨網域連線，伺服器端在送出 preflight response 時，就必須要帶有特定的 header。使用 rack-cors 這個 gem 來處理 CORS，它的用法也很簡單，只要在 config 中加入相關的設定就可以了：config/application.rb

參考資料:

1.https://sibevin.github.io/posts/2017-06-05-101518-note-cors

2. https://developer.mozilla.org/zh-TW/docs/Web/HTTP/CORS

3. https://developer.mozilla.org/en-US/docs/web/http/cors#simple_requests