Server REST con esp8266 e esp32: richieste CORS, OPTION e GET – Parte 4

DI RENZO MISCHIANTI · PUBBLICATO 15 LUGLIO 2020 · AGGIORNATO 5 NOVEMBRE 2023

Ora, abbiamo fatto chiamate delle chiamate dal programma Postman, ma fare lo stesso da un browser web non è la stessa cosa.

REST server on esp8266 and esp32 CORS request OPTION and GET

Ci sono alcune politiche di sicurezza che non possiamo ignorare. La più importante/fastidiosa è che se si desidera effettuare una chiamata REST a un server da un client, con dominio diverso (o origine se si preferisce) si entra nel tunnel dei CORS.

Cross-Origin Resource Sharing (CORS) è un meccanismo che utilizza intestazioni HTTP aggiuntive per indicare ai browser di fornire a un'applicazione Web in esecuzione su un'origine, l'accesso a risorse selezionate da un'origine diversa. Un'applicazione Web esegue una ***richiesta HTTP tra origini*** *quando richiede una risorsa che ha un'origine diversa (dominio, protocollo o porta) dalla propria. (Cit.)*

MA che significa? 😛

Se prendi questo codice html/js

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Do get request</title>
</head>
<body>
    <div>Normal response</div>
    <div id="content"></div>
    <div>Error response</div>
    <div id="errorContent"></div>
</body>
<script>
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            // Typical action to be performed when the document is
ready:
            document.getElementById("content").innerHTML =
xhttp.responseText;
        }
    };
    xhttp.onerror = function () {
        document.getElementById("errorContent").innerHTML =
"Status code is " + this.status + " click F12 and check what is
```

```
the problem on console";
    };

    xhttp.open("GET", "http://esp8266/settings", true);
    xhttp.send();

</script>
</html>
```

e provi ad eseguire questa semplice GET

```
    xhttp.open("GET", "http://esp8266/settings", true);
```

Il tuo browser notifica sulla console (F12 per vederla) che tu hai un

```
Access to XMLHttpRequest at 'http://esp8266/settings' from origin
'null' has been blocked by CORS policy: No 'Access-Control-Allow-
Origin' header is present on the requested resource.
```

errore CORS, questo perché stai tentando di fare una chiamata ad un end-point REST da un'origine differente.

Gestire una chiamata CORS in GET

Server REST con esp8266 ed esp32 il tunnel dei CORS

Il tuo primo tentativo è quello di aggiungere l'intestazione per specificare che il server deve accettare tutte le origini e i verbi, quindi è possibile aggiungere dopo l'`open` e prima della `send`

```
    xhttp.setRequestHeader('Access-Control-Allow-Headers', '*');
    xhttp.setRequestHeader('Access-Control-Allow-Origin', '*');
```

Ma ora abbiamo un nuovo errore

```
OPTIONS http://esp8266/settings 404 (Not Found)

Access to XMLHttpRequest at 'http://esp8266/settings' from origin
'null' has been blocked by CORS policy: Response to preflight
request doesn't pass access control check: No 'Access-Control-
Allow-Origin' header is present on the requested resource.
```

Qui puoi verificare che il client non tenti di eseguire la richiesta GET ma un OPTION e di conseguenza non trova quell'end point.

E l'errore specifica che "`Response to preflight request doesn't pass access control check`", e questo è correlato al precedente 404 su OPTION.

Nelle chiamate CORS, con il verbo `OPTIONS` viene inviata una richiesta di verifica preliminare, in modo che il server possa rispondere se è accettabile inviare la richiesta con questi parametri.

E ora aggiungeremo l'endpoint OPTION al server REST.

E lo aggiungeremo allo sketch esp8266

```
void sendCrossOriginHeader(){
        server.send(204);
}
```

```
// Define routing
void restServerRouting() {
    server.on("/", HTTP_GET, []() {
        server.send(200, F("text/html"),
            F("Welcome to the REST Web Server"));
    });
    server.on(F("/helloWorld"), HTTP_GET, getHelloWord);
    server.on(F("/settings"), HTTP_OPTIONS,
sendCrossOriginHeader);
    server.on(F("/settings"), HTTP_GET, getSettings);
}
```

Ma otteniamo ancora un errore

```
Access to XMLHttpRequest at 'http://esp8266/settings' from origin
'http://localhost:63342' has been blocked by CORS policy: No
'Access-Control-Allow-Origin' header is present on the requested
resource.
```

Il problema è che l'OPTION, anche se, ha una risposta in 204 (tutti i 2xx sono risposte positive) non restituisce l'intestazione corretta, quindi aggiungeremo l'intestazione alla risposta del GET.

```
        httpRestServer.sendHeader(F("Access-Control-Allow-
Origin"), F("*"));
        httpRestServer.sendHeader(F("Access-Control-Max-Age"),
F("600"));
        httpRestServer.sendHeader(F("Access-Control-Allow-
Methods"), F("PUT,POST,GET,OPTIONS"));
        httpRestServer.sendHeader(F("Access-Control-Allow-
Headers"), F("*"));
```

E ora finalmente

```
Normal response
{"ip":"192.168.1.129","gw":"192.168.1.1","nm":"255.255.255.0"}
Error response
```

Ok, abbiamo finalmente una chiamata con verbo GET in CORS.

Per esp32 devi solo cambiare questi include

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
```

in

```
#include <WiFi.h>
#include <WebServer.h>
#include <ESPmDNS.h>
```

e le informazioni sul chip così

```
       doc["chipRevision"] = ESP.getChipRevision();
       doc["flashChipMode"] = ESP.getFlashChipMode();
       doc["flashChipSize"] = ESP.getFlashChipSize();
       doc["flashChipSpeed"] = ESP.getFlashChipSpeed();
```

Ecco lo sketch completo

```
/*
 *  Json parametric GET REST response with ArduinoJSON library
 *  by Mischianti Renzo <https://mischianti.org>
```

```
 *
 *  https://mischianti.org/
 *
 */

#include "Arduino.h"
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <ArduinoJson.h>

const char* ssid = "<YOUR-SSID>";
const char* password = "<YOUR-PASSWD>";

ESP8266WebServer server(80);

void setCrossOrigin(){
        server.sendHeader(F("Access-Control-Allow-Origin"),
F("*"));
        server.sendHeader(F("Access-Control-Max-Age"), F("600"));
        server.sendHeader(F("Access-Control-Allow-Methods"),
F("PUT,POST,GET,OPTIONS"));
        server.sendHeader(F("Access-Control-Allow-Headers"),
F("*"));
};


// Serving Hello world
void getHelloWord() {
          DynamicJsonDocument doc(512);
          doc["name"] = "Hello world";

          Serial.print(F("Stream..."));
          String buf;
          serializeJson(doc, buf);
          server.send(200, "application/json", buf);
          Serial.print(F("done."));
}

// Serving Hello world
void getSettings() {
        setCrossOrigin();
//
          // Allocate a temporary JsonDocument
          // Don't forget to change the capacity to match your
requirements.
          // Use arduinojson.org/v6/assistant to compute the
capacity.
        //  StaticJsonDocument<512> doc;
          // You can use DynamicJsonDocument as well
          DynamicJsonDocument doc(512);
```

```
            doc["ip"] = WiFi.localIP().toString();
            doc["gw"] = WiFi.gatewayIP().toString();
            doc["nm"] = WiFi.subnetMask().toString();

            if (server.arg("signalStrength")== "true"){
                    doc["signalStrengh"] = WiFi.RSSI();
            }

            if (server.arg("chipInfo")== "true"){
                    doc["chipId"] = ESP.getChipId();
                    doc["flashChipId"] = ESP.getFlashChipId();
                    doc["flashChipSize"] = ESP.getFlashChipSize();
                    doc["flashChipRealSize"] =
ESP.getFlashChipRealSize();
            }
            if (server.arg("freeHeap")== "true"){
                    doc["freeHeap"] = ESP.getFreeHeap();
            }

            Serial.print(F("Stream..."));
            String buf;
            serializeJson(doc, buf);
            server.send(200, F("application/json"), buf);
            Serial.print(F("done."));
}

void sendCrossOriginHeader(){
        Serial.println(F("sendCORSHeader"));
    setCrossOrigin();
        server.send(204);
}

// Define routing
void restServerRouting() {
    server.on("/", HTTP_GET, []() {
        server.send(200, F("text/html"),
            F("Welcome to the REST Web Server"));
    });
    server.on(F("/helloWorld"), HTTP_GET, getHelloWord);
    server.on(F("/settings"), HTTP_OPTIONS,
sendCrossOriginHeader);
    server.on(F("/settings"), HTTP_GET, getSettings);

}

// Manage not found URL
void handleNotFound() {
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
```

```
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " " + server.argName(i) + ": " + server.arg(i) +
"\n";
  }
  server.send(404, "text/plain", message);
}

void setup(void) {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  // Activate mDNS this is used to be able to connect to the
server
  // with local DNS hostmane esp8266.local
  if (MDNS.begin("esp8266")) {
    Serial.println("MDNS responder started");
  }

  // Set server routing
  restServerRouting();
  // Set not found response
  server.onNotFound(handleNotFound);
  // Start server
  server.begin();
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
}
```

Grazie

1. Server REST su esp8266 o esp32: introduzione
2. Server REST su esp8266 o esp32: GET e formattazione JSON
3. Server REST su esp8266 o esp32: POST, PUT, PATCH, DELETE
4. Server REST su esp8266 o esp32: richieste CORS, OPTION e GET
5. Server REST su esp8266 o esp32: richieste CORS, OPTION e POST

Etichette: esp32 esp8266 REST API Wemos D1 mini

LASCIA UN COMMENTO

Login with your Social ID

I agree to my personal data being stored and used as per Privacy Policy

Commento *

Nome * **Email ***

Sito web

Subscribe newsletter!

Invia commento