Server REST con esp8266 e esp32: richieste CORS, OPTION e POST – Parte 5

DI RENZO MISCHIANTI · PUBBLICATO 26 LUGLIO 2020 · AGGIORNATO 5 NOVEMBRE 2023

Come ho scritto nel precedente articolo 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.

Server REST con esp8266 e esp32: CORS request, OPTION e POST

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.)*

Nell'ultimo articolo ho mostrato come è possibile bypassare la politica CORS per il GET, ora andremo a vedere come una POST abbia bisogno di ulteriori informazioni.

Fare riferimento all'articolo precedente per lo sketch di base da modificare.

Gestire una chiamata CORS in POST

Come abbiamo fatto per il GET andremo a gestire una POST.

Ho creato un semplice file html con una POST.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Do POST 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 == 201) {
            // 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";
    };

    var params = {
        ip: "192.168.1.123",
        gw: "192.168.1.1",
        nm: "192.168.1.255"
    }

    xhttp.open("POST", "http://192.168.1.129/settings", true);

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

    xhttp.setRequestHeader('Content-type', 'application/json')
    xhttp.send(JSON.stringify(params)) // Make sure to stringify

</script>
```

```
</html>
```

E ora vado a creare la POST con tutte le considerazioni precedenti.

Qui il routing

```
// 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);

    server.on(F("/settings"), HTTP_POST, setSettings);
}
```

Qui la setSettings POST consumer.

```
void setSettings() {
        // expected
        //
{"ip":"192.168.1.129","gw":"192.168.1.1","nm":"255.255.255.0"}
        Serial.println(F("postConfigFile"));

        setCrossOrigin();

    String postBody = server.arg("plain");
    Serial.println(postBody);

        DynamicJsonDocument doc(512);
        DeserializationError error = deserializeJson(doc,
postBody);
        if (error) {
                // if the file didn't open, print an error:
                Serial.print(F("Error parsing JSON "));
                Serial.println(error.c_str());

                String msg = error.c_str();

                server.send(400, F("text/html"), "Error in parsin
```

```
json body! <br>"+msg);

        }else{
                JsonObject postObj = doc.as<JsonObject>();

                Serial.print(F("HTTP Method: "));
                Serial.println(server.method());

        if (server.method() == HTTP_POST) {
            if ((postObj.containsKey("ip"))) {

                Serial.print(F("Open config file..."));
                // Here you can open your file to store your
config
                // Now I simulate It and set configFile a true
                bool configFile = true;
                if (!configFile) {
                    Serial.println(F("fail."));
                    server.send(304, F("text/html"), F("Fail to
store data, can't open file!"));
                }else{
                        Serial.println(F("done."));
                                const char* address =
postObj[F("ip")];
                                const char* gatway =
postObj[F("gw")];
                                const char* netMask =
postObj[F("nm")];

                        Serial.print("ip: ");
                        Serial.println(address);
                        Serial.print("gw: ");
                        Serial.println(gatway);
                        Serial.print("nm: ");
                        Serial.println(netMask);

//                      server.sendHeader("Content-Length",
String(postBody.length()));
                        server.send(201, F("application/json"),
postBody);

//                      Serial.println(F("Sent reset page"));
//                                      delay(5000);
//                                      ESP.restart();
//                                      delay(2000);
                }
            }
            else {
                server.send(204, F("text/html"), F("No data found,
or incorrect!"));
            }
        }
```

```
    }
}
```

E ora andremo ad eseguire la POST, ma avremo una sorpresa.

```
index.html?_ijt=j45g17rditk4jvsopqbsb6mohs:1 Access to
XMLHttpRequest at 'http://192.168.1.129/settings' from origin
'http://localhost:63342' 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.
```

Si è verificato un errore CORS, poiché la richiesta GET richiede l'intestazione CORS solo nella risposta GET, il POST va abilitato da una OPTION, perciò andremo ad aggiungere l'intestazione CORS nella OPTION.

```
void sendCrossOriginHeader(){
        Serial.println(F("sendCORSHeader"));
        server.sendHeader(F("access-control-allow-credentials"),
F("false"));
        setCrossOrigin();
        server.send(204);
}
```

Ora se riproviamo avremo la risposta che ci aspettiamo.

```
Normal response
{"ip":"192.168.1.123","gw":"192.168.1.1","nm":"192.168.1.255"}
Error response
```

Ecco lo sketch completo

Per l'esp32 devi 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 come segue

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

```
/*
 *      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 setSettings() {
        // expected
        //
{"ip":"192.168.1.129","gw":"192.168.1.1","nm":"255.255.255.0"}
        Serial.println(F("postConfigFile"));

        setCrossOrigin();

    String postBody = server.arg("plain");
    Serial.println(postBody);

        DynamicJsonDocument doc(512);
        DeserializationError error = deserializeJson(doc,
postBody);
        if (error) {
                // if the file didn't open, print an error:
                Serial.print(F("Error parsing JSON "));
                Serial.println(error.c_str());

                String msg = error.c_str();

                server.send(400, F("text/html"), "Error in parsin
json body! <br>"+msg);

        }else{
                JsonObject postObj = doc.as<JsonObject>();

                Serial.print(F("HTTP Method: "));
                Serial.println(server.method());

        if (server.method() == HTTP_POST) {
            if ((postObj.containsKey("ip"))) {

                Serial.print(F("Open config file..."));
                // Here you can open your file to store your
```

```
config
                // Now I simulate It and set configFile a true
                bool configFile = true;
                if (!configFile) {
                    Serial.println(F("fail."));
                    server.send(304, F("text/html"), F("Fail to
store data, can't open file!"));
                }else{
                        Serial.println(F("done."));
                                const char* address =
postObj[F("ip")];
                                const char* gatway =
postObj[F("gw")];
                                const char* netMask =
postObj[F("nm")];

                        Serial.print("ip: ");
                        Serial.println(address);
                        Serial.print("gw: ");
                        Serial.println(gatway);
                        Serial.print("nm: ");
                        Serial.println(netMask);

//                      server.sendHeader("Content-Length",
String(postBody.length()));
                        server.send(201, F("application/json"),
postBody);

//                      Serial.println(F("Sent reset page"));
//                                          delay(5000);
//                                          ESP.restart();
//                                          delay(2000);
                }
            }
            else {
                server.send(204, F("text/html"), F("No data found,
or incorrect!"));
            }
        }
    }
}


void sendCrossOriginHeader(){
        Serial.println(F("sendCORSHeader"));

        server.sendHeader(F("access-control-allow-credentials"),
F("false"));

        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);

    server.on(F("/settings"), HTTP_POST, setSettings);
}

// 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