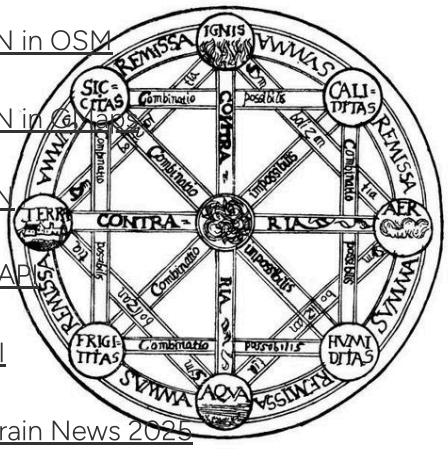


Is the Artist a Myth?

You think creativity is personal genius? Five thinkers prove it's a shared process beyond the self.

Discover the shift

about

Fleischmann 7232 N

Commercial QR-Code Platform Learnings

This will be the preparing for maXbox Starter Tutorial 193.

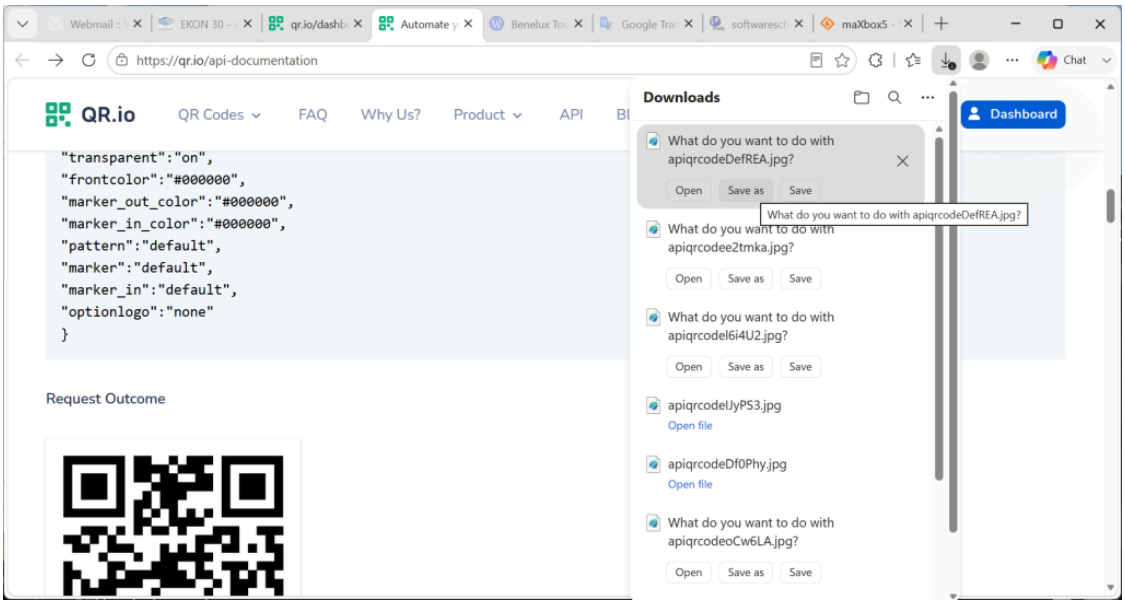
As an example we look behind a API which generates QR-Codes. We compare 2 solutions, one is straightforward and the second refactored solution with some pitfalls and notes to keep in mind. By using QR.io you will be able to keep track of how many people scan your QR Codes, from where and on what date the call is.

Also, for those non-developers, you can create fully customized landing pages for your QR Codes in a dashboard. No Coding Required! But using the API will need some code template:

```
{$I .\NINJAAPIKEY.INC}

const
   QR_APIKEY = QRC_APIKEY;

function Get_QRCodelink_API_Post(Sender: TObject; aURL: strin
var httpreq: THttpRequestC;
    JSONStr: string;  jo: TJSON;
  begin
```

```
      httpreq:= THttpRequestC.create(self);
      //https://api.qr.io/v1/create
      //httpreq.headers.add('Content-type: application/json; c
       httpreq.headers.add('"Content-type": "application/json'
       httpreq.headers.add('User-Agent:'+USERAGENT5);
      try
        //JSONStr:= GeoJSON.Stringify; //ToJSON;
        jo:= TJSON.Create();
        JSONStr:= '{' +
          '"apikey":"'+QR_APIKEY+'",'+
          '"data":"https://qr.io",'+
          '"transparent":"on",'+
          '"fontcolor":"#000000",'+
          '"marker_out_color":"#000000",'+
          '"marker_in_color":"#000000",'+
        //'"prompt": "%s",'+
          '"pattern":"default",'+
          '"marker":"default",'+
          '"marker_in":"default",'+
          '"optionlogo":"none"'+
          // '"api_key": ""'+
          '}';

        Memo2.Lines.Add('Post JSON created: ' + JSONStr);
        try
          writeln(formatJson(JSONStr));
        except
          write('JSON FormatError: '+ExceptiontoString(except:
        end;
        //writ(botostr(httpreq.Post2(TBASE_URL6,JSONStr)))
        if httpreq.Post2(aURL,JSONStr) then begin
          result:= (httpreq.Response.contentasstring);
          jo.parse(result)
          writ('contentasjpg: '+jo.values['jpg'].asstring);
          result:= jo.values['jpg'].asstring;
          writ('content-length:'+itoa(httpreq.response.Content
        end;
      except
      //on E: Exception do
        write('REQC_Err: '+ExceptiontoString(exceptiontype,exc
      finally
        httpreq.Free;  //THttpRequestC;
        jo.free;
      end;
    end;

openWeb(Get_QRCodelink_API_Post(self, 'https://api.qr.io/v1/c
```

Then we got something like this: contentasjpg: https://qr.io/jpg-download/CK7li9/api-qr-code-zzK7li9



https://sourceforge.net/projects/maxbox5/files/examples/1493_qrcode_api1.txt/download

What it does

`Get_QRCodelink_API_Post` sends a JSON `POST` request to the QR.io API, asks it to generate a QR code for `https://qr.io/`, extracts the returned JPG link, and returns that link as a string.

The QR.io endpoint is documented as a `POST` endpoint whose request body is a JSON object containing fields such as `apikey`, `data`, color settings, transparency, and pattern options.[qr]

The final line:

```
openWeb(
  Get_QRCodelink_API_Post(
    Self,
    'https://api.qr.io/v1/create'
  )
);
```

therefore appears to open the generated QR-code image or URL in a browser or related viewer.

Execution flow

1. It creates an HTTP request object: `httpreq:= THttpRequestC.Create(Self);`
2. It adds two HTTP headers:`httpreq.Headers.Add('"Content-type": "application/json"'); httpreq.Headers.Add('User-Agent:' +`

`USERAGENT5);`

3. It constructs a JSON request body containing:`{ "apikey": "...", "data": "https://qr.io/", "transparent": "on", "fontcolor": "#000000", "marker_out_color": "#000000", "marker_in_color": "#000000", "pattern": "default", "marker": "default", "marker_in": "default", "optionlogo": "none" }`
4. It writes the generated JSON to `Memo2`.
5. It attempts to format or validate the JSON using:`writeln(formatJson(JSONStr));`
6. It posts the JSON to the URL supplied in `aURL`:`if httpreq.Post2(aURL, JSONStr) then`
7. If the request succeeds, it reads the HTTP response as text:`result := httpreq.Response.ContentAsString;`
8. It parses that response as JSON:`jo.Parse(result);`
9. It retrieves the response's `jpg` property:`result := jo.Values['jpg'].AsString;`
10. It logs the response content length and frees the HTTP and JSON objects.

In simplified pseudocode:

```
create HTTP client
set JSON-related headers
build QR.io JSON request
POST request to aURL
parse response JSON
return response["jpg"]
free resources
```

Important problems and fix

This 7 tips to call a foreign API

You have to keep in mind the following 7 notes to handle with a API as Post with JSON:

1. The `Content-Type` header is malformed

This line:

```
httpreq.Headers.Add('"Content-type": "application/json"');
```

is not a valid normal HTTP header. It looks like a JSON property rather than a header declaration.

It should usually be something like:

```
httpreq.Headers.Add('Content-Type: application/json; charset=UT
```

or, depending on the HTTP library:

```
httpreq.Headers.Add('Content-Type', 'application/json; charset=
```

The commented-out version is much closer to correct:

```
// httpreq.Headers.Add('Content-type: application/json; charset
```

2. The function ignores its `Sender` parameter

The declaration contains:

```
Sender: TObject
```

but the function never uses `Sender`. It uses `Self` instead:

```
httpreq := THttpRequestC.Create(Self);
```

If this is a standalone function, `Self` may not be valid or may refer to an enclosing method's object rather than `Sender`. If the intention is to use the caller as the owner, this should probably be:

```
httpreq := THttpRequestC.Create(Sender);
```

Alternatively, remove `Sender` from the parameter list if it is unnecessary.

3. The JSON is manually concatenated

This part is fragile:

```
JSONStr := '{' +
  '"apikey":"' + QR_APIKEY + '",' +
  '"data":"https://qr.io/",' +
  ...
```

If `QR_APIKEY` ever contains a quote, backslash, or another character requiring JSON escaping, the resulting JSON becomes invalid. A JSON library should be used to construct the object rather than concatenating strings.

Also, `jo` is used only for parsing the response; it is not used to construct `JSONStr`.

4. The response may not contain `jpg`

The code assumes that the API response is valid JSON and contains a field named `jpg`:

```
result := jo.Values['jpg'].AsString;
```

If the API returns an error, HTML, malformed JSON, or a response with a different property name, this can raise an exception or return an empty value. The code should check:

- the HTTP status code;
- whether the response is valid JSON;
- whether `jpg` exists;
- whether the value is non-empty.

For example, conceptually:

```
if not jo.Contains('jpg') then
  raise Exception.Create('QR.io response does not contain "jpg"
```

The exact method depends on the JSON library.

5. The API key is placed in the request body

The code sends:

```
"apikey": "..."
```

That matches the QR.io documentation shown in the search results, where the API key is included as a JSON request parameter. Nevertheless, the API key should not be written to a memo or log. Since this code logs the complete JSON:[qr]

```
Memo2.Lines.Add('Post JSON created: ' + JSONStr);
```

it may expose the API key in the user interface, debug logs, or diagnostic files.

A safer version would mask it:

```
Memo2.Lines.Add('Post JSON created: API key omitted');
```

6. The function can return an undefined or stale result

If `Post2` returns `False`, the function does not explicitly assign `Result`. In Delphi, this can result in an empty string for a managed string result, but relying on that is poor practice. Initialize it explicitly:

```
Result := '';
```

at the beginning.

7. `jo.Free` may be unsafe if construction fails

The code creates `jo` inside the `try` block:

```
jo := TJSON.Create();
```

but frees it in the `finally` block:

```
jo.Free;
```

If `TJSON.Create` itself raises an exception, `jo` may be uninitialized. In modern Delphi, local managed variables should normally be initialized, but it is still clearer and safer to initialize object references:

```
httpreq := nil;
jo := nil;
try
  httpreq := THttpRequestC.Create(Sender);
  jo := TJSON.Create;
finally
  jo.Free;
  httpreq.Free;
end;
```

A cleaner version

The following keeps the same general design, though the exact JSON methods depend on the libraries in use:

```
function Get_QRCodelink_API_Post(Sender: TObject; const AURL: s
var
  HttpReq: THttpRequestC;
```

```
  JSONStr: string;
  JO: TJSON;
  ResponseText: string;
begin
  Result := '';
  HttpReq := nil;
  JO := nil;

  try
    HttpReq := THttpRequestC.Create(Sender);

    HttpReq.Headers.Add(
      'Content-Type: application/json; charset=UTF-8'
    );
    HttpReq.Headers.Add(
      'User-Agent: ' + USERAGENT5
    );

    JSONStr :=
      '{' +
      '"apikey":"' + EscapeJson(QR_APIKEY) + '",' +
      '"data":"https://qr.io/",' +
      '"transparent":"on",' +
      '"frontcolor":"#000000",' +
      '"marker_out_color":"#000000",' +
      '"marker_in_color":"#000000",' +
      '"pattern":"default",' +
      '"marker":"default",' +
      '"marker_in":"default",' +
      '"optionlogo":"none"' +
      '}';

    if not HttpReq.Post2(AURL, JSONStr) then
      Exit;

    ResponseText := HttpReq.Response.ContentAsString;

    JO := TJSON.Create;
    JO.Parse(ResponseText);

    if JO.Values['jpg'] <> nil then
      Result := JO.Values['jpg'].AsString;
  finally
    JO.Free;
    HttpReq.Free;
  end;
```

```
end;
```

Two details require verification in your codebase:

- QR.io's documentation uses `frontcolor`, whereas your code sends `fontcolor`. The documented example uses `"frontcolor":"#000000"`, so `fontcolor` may be a typo.[qr][qr]
- The response property must really be named `jpg`; confirm that against the actual API response or current API documentation before relying on it.

In short, the function is intended to generate a customized QR code through QR.io and return the URL of the generated JPG, but it has a malformed `Content-Type` header, a likely `fontcolor` typo, unsafe JSON construction, and insufficient response/error checking.

```
function Get_QRCodelink_API_Post2(Sender: TObject; const AURI
var
  HttpReq: THttpRequestC;
  JSONStr,ResponseText: string;
  JO: TJSON;
 begin
  Result:= '';
  HttpReq:= nil;
  JO:= nil;

  try
    HttpReq:= THttpRequestC.Create(Self);
    HttpReq.Headers.Add(
      'Content-Type: application/json; charset=UTF-8'
    );
    HttpReq.Headers.Add(
      'User-Agent: ' + USERAGENT5
    );

    JSONStr:=
      '{' +
      '"apikey":"' + McJsonEscapeString(QR_APIKEY,jetNormal)
      '"data":"https://qr.io/",' +
      '"transparent":"on",' +
      '"frontcolor":"#000000",' +
      '"marker_out_color":"#000000",' +
      '"marker_in_color":"#000000",' +
      '"pattern":"default",' +
      '"marker":"default",' +
      '"marker_in":"default",' +
      '"optionlogo":"none"' +
      '}';

    if not HttpReq.Post2(AURL, JSONStr) then
```

```
35        Exit;
36
37      ResponseText:= HttpReq.Response.ContentAsString;
38
39      JO:= TJSON.Create;
40      JO.Parse(ResponseText);
41
42      if JO.Values['jpg'] <> nil then
43        Result:= JO.Values['jpg'].AsString;
44    finally
45      JO.Free;
46      HttpReq.Free;
47    end;
48  end;
```

E10 Rheingold

Advertisements

Occasionally, some of your visitors may see an advertisement here, as well as a Privacy & Cookies banner at the bottom of the page. You can hide ads completely by upgrading to one of our paid plans.

UPGRADE NOW DISMISS MESSAGE

Posted in Code, Trains

Leave a comment