DEVCON 2024
EUROPEAN DELPHI CONFERENCE

CORS a ostacoli?
Metti in sicurezza le risorse delle tue Web API in Delphi

Marco Breveglieri

ROMA - 14 E 15 NOVEMBRE 2024

Marco Breveglieri

👨‍💻 Software Developer | 👨‍🏫 Trainer and Consultant | 📽️ Tech Content Creator | 💜 Community lover

Homepage
https://www.breveglieri.it

Blog tecnico
https://www.compilaquindiva.com

Delphi Podcast
https://www.delphipodcast.com

Canale Twitch
https://twitch.tv/compilaquindiva

Tutti gli altri link: **https://linktr.ee/marco.breveglieri**

Prima di iniziare...

...un piccolo sondaggio.

Ti è mai capitato questo errore?

```
Access to fetch at 'https://joke-api-strict-cors.appspot.com/r localhost/:1
andom joke' from origin 'http://localhost:3000' has been blocked by CORS
policy: No 'Access-Control-Allow-Origin' header is present on the requested
resource. If an opaque response serves your needs, set the request's mode
to 'no-cors' to fetch the resource with CORS disabled.
```

Dimmi la verità... 😉

...e se non è successo a te...

James Eastham
@plantpowerjames · Follow

CORS has to be the single must frustrating part of any development I do. I've solved it so many times and yet every time I build an API I still end up down a Google rabbit hole 🤦‍♂️

12:31 PM · May 22, 2022

7 Reply Copy link

Read 3 replies

Ben
@digitaltrouble · Follow

People wonder what it takes to be a senior developer. One CORS issue. By the time you've fixed it, you will be a senior in any way possible... 😑

12:11 PM · Apr 9, 2021

Reply Copy link

Read 3 replies

Void ⚡
@codewithvoid · Follow

If you have never faced a CORS issue, are you even a web developer?

10:33 AM · Oct 16, 2022

864 Reply Copy link

Read 92 replies

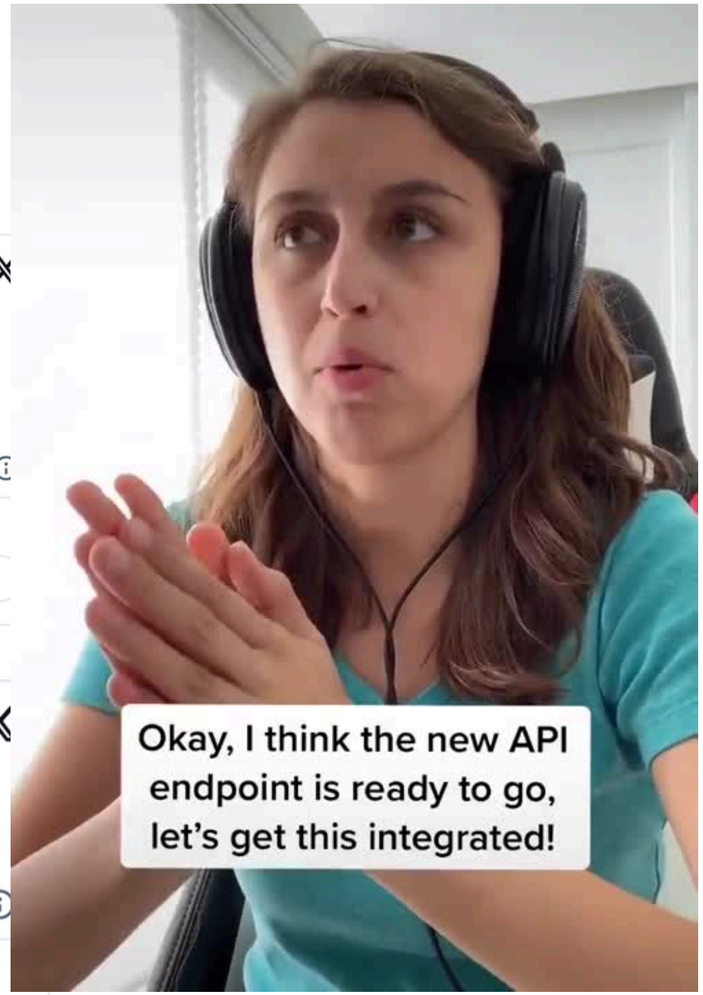
Developers Swearing
@gitlost · Follow
Automated

Fuck cors

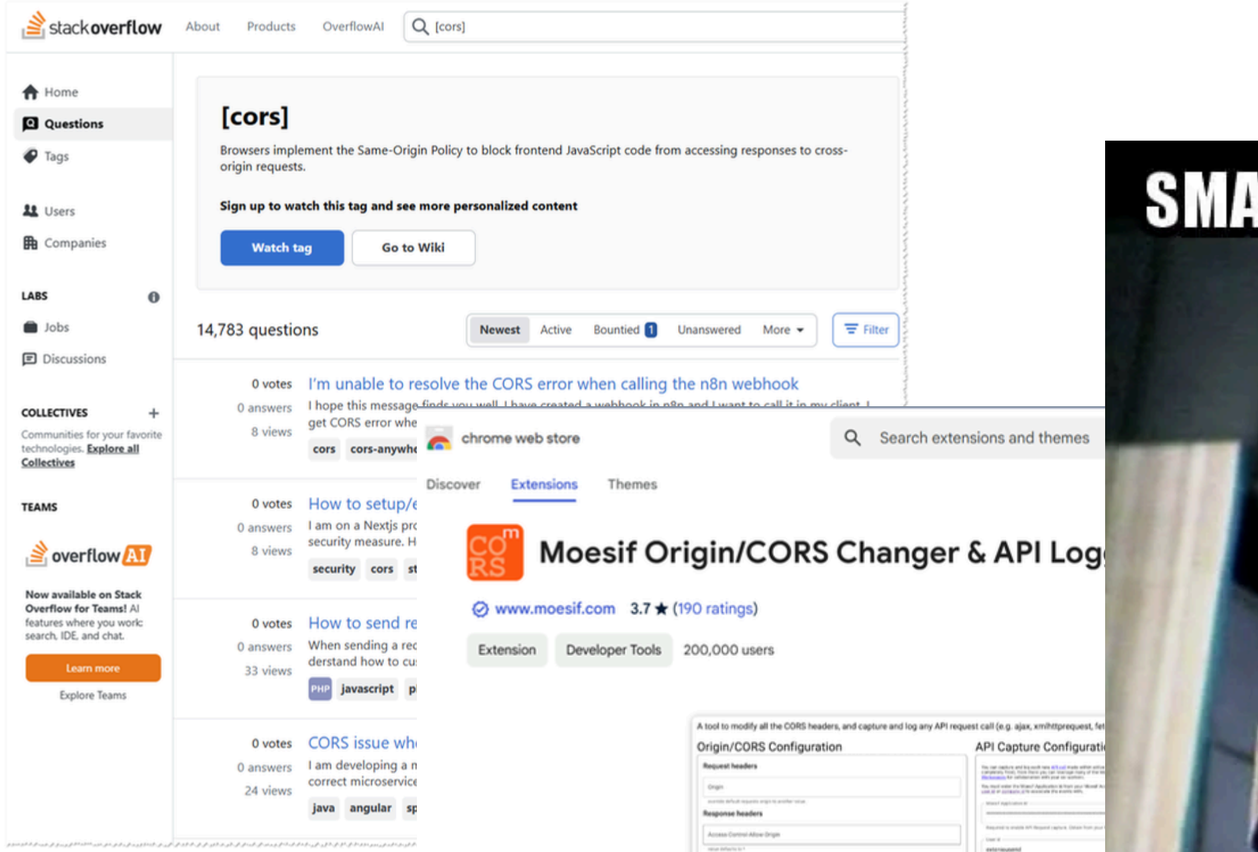
2:32 AM · Jul 14, 2017

3 Reply Copy link

Read more on X



StackOverflow, aiutami tu!



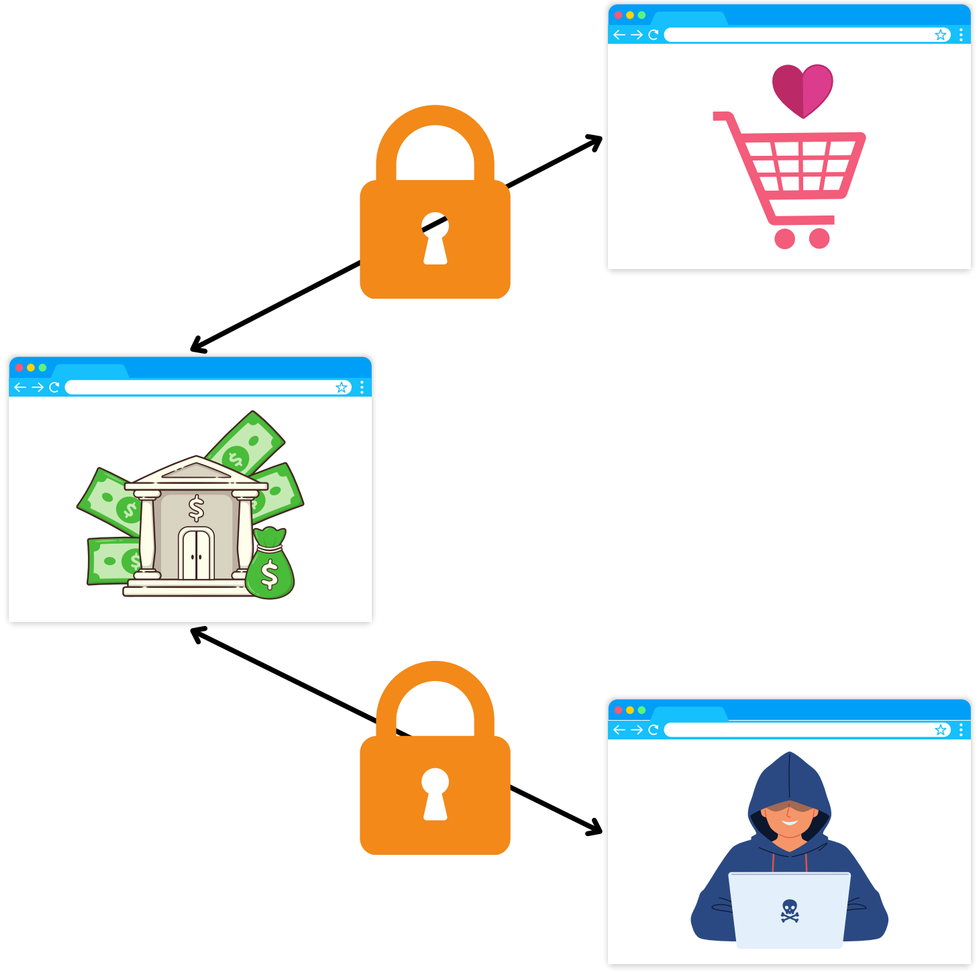
“SOP… opera”

Parliamo della Same-Origin Policy (SOP)

Immaginiamo che...

Same-Origin Policy (SOP)

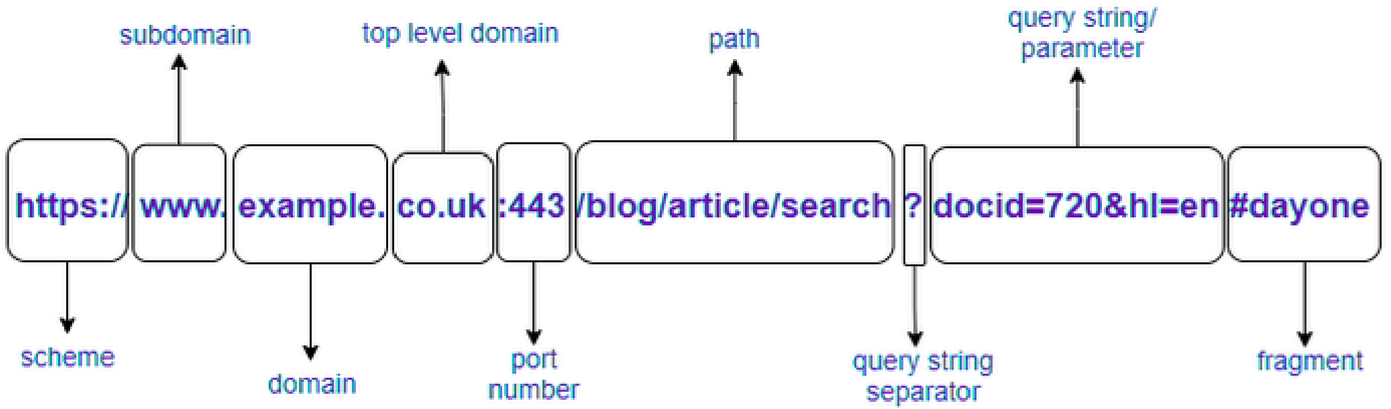
- E’ una restrizione forzata da tutti i **browser**
- E’ attiva per default
- Regola l’accesso ai dati in base all’**origine**
- Non richiede alcuna configurazione lato server
- Non previene la “scritture”, bensì la **lettura** dei dati



Definiamo una “Origin” /1

L’origine di una risorsa
(**Origin**) è rappresentata da

- **Scheme** (protocollo)
- **Host Name** (dominio)
- **Port** (porta)



https://www.geeksforgeeks.org/components-of-a-url/

usati per accedervi tramite il
relativo URL.

Definiamo una “Origin” /2

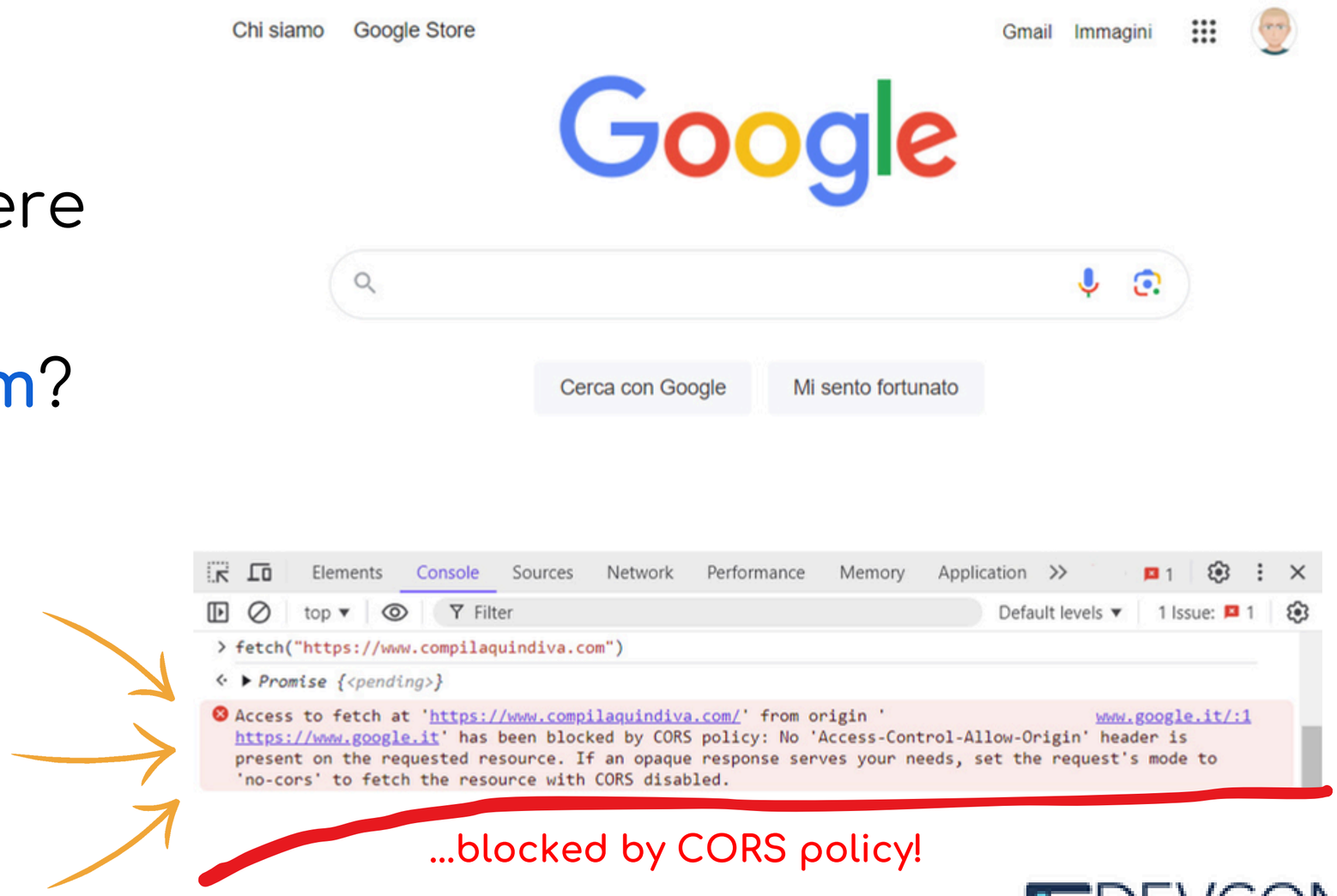
Consideriamo questo URL:

https://www.compilaquindiva.com/blog/

URL	SOP 👌 Ok?	Perché?
https://www.compilaquindiva.com/	✔️	Stesso protocollo, dominio e porta
https://www.compilaquindiva.com/about/	✔️	Stesso protocollo, dominio e porta
http://www.compilaquindiva.com/	❌	Diverso protocollo, diversa porta
https://blog.compilaquindiva.com/	❌	Diverso dominio
https://www.compilaquindiva.com:8080/	❌	Diversa porta

SOP in azione!

Cosa succede se apriamo google.it e cerchiamo di accedere a una risorsa di compilaquindiva.com?



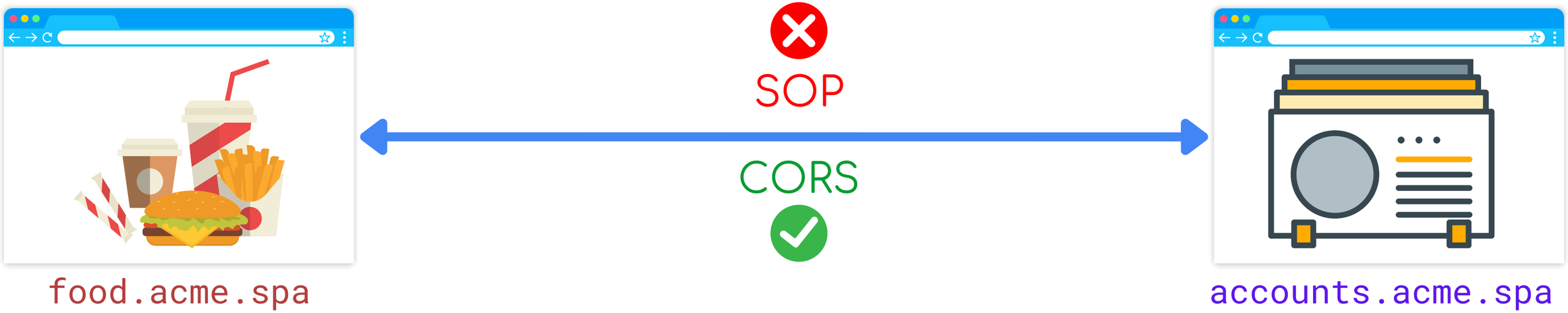
Come risolviamo?

CORS

Cross-Origin Resource Sharing

Che cos'è CORS?

Cross-Origin Resource Sharing (per gli amici "CORS") è un meccanismo che usa uno o più header HTTP per definire le origini alle quali il browser deve "permettere" l'accesso.



Come si configura CORS?

- Gli sviluppatori configurano le “regole CORS” direttamente nel codice dell’applicazione Web
- Le modalità di configurazione dipendono da librerie e framework adottati
  - Possono essere “globali” o “per richiesta”
  - Possono fare uso di “middleware” o “attributi”

```
// Middleware
FEngine.AddMiddleware(TMVCCORSMiddleware.Create)
.
```

Request HTTP:
chiedere è lecito.

Intestazione “Origin”

Ad ogni richiesta HTTP, **il browser aggiunge l’intestazione** Origin.

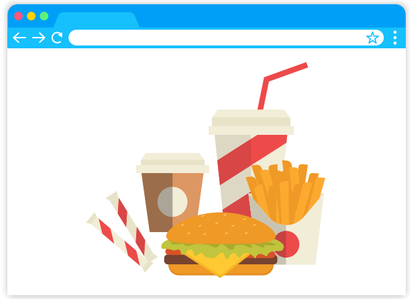
- Sostituisce il più celebre “Referrer”
  - E’ obsoleto: non è usabile a questo scopo
  - Viene bloccato da browser e tool anti-tracing
- Lo troviamo in tante altre richieste (ad esempio, Web Socket)
- Talvolta ha comportamenti difformi
  - In caso di GET “no-cors” è assente
  - In caso di POST “no-cors” è presente (forzato)

```
Origin: https://developer.mozilla.org:80
```

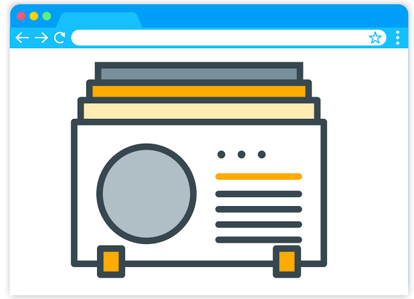
Response HTTP:
rispondere è cortesia.

Access-Control-Allow-Origin /1

E' l'intestazione che indica se la risposta può essere condivisa con il codice che ne ha fatto richiesta dall'origine specificata.



food.acme.spa



accounts.acme.spa

```
GET /home.aspx HTTP/1.1
Host: accounts.acme.spa
Origin: food.acme.spa
...
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: food.acme.spa
...
```

Access-Control-Allow-Origin /2

Sono ammesse **n. 3 tipologie di valori diversi**:

```
Access-Control-Allow-Origin: *
Access-Control-Allow-Origin: origin
Access-Control-Allow-Origin: null
```

Attenzione!

- Il valore "*" esclude la modalità "with credentials"
- Se è indicata un'origine specifica, deve essere unica
- Non si possono usare wildcard con la singola origine
- Il nome dell'header è "case insensitive", il valore no!
- Se la risorsa non contiene nulla di privato, usa pure "*"

Access-Control-Allow-Origin /3

Se consentito, l'origine può accedere al "response body" e ai valori di alcuni specifici header:

- `Cache-Control`
- `Content-Language`
- `Content-Length`
- `Content-Type`
- `Expires`
- `Last-Modified`
- `Pragma`

Access-Control-Expose-Headers

Consente all'origine di **ottenere l'accesso ad altre intestazioni**, che sono precluse per default.

```
Access-Control-Expose-Headers: *
Access-Control-Expose-Headers: Header1, Header2, ...
```

Attenzione!

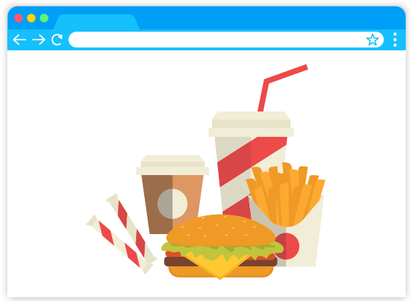
- **I valori sono "case insensitive"**
  - Sono tali i nomi degli header menzionati
  - E' trattato diversamente da "Origin"
- **Alcuni header restano nascosti, per motivi di sicurezza**
  - "Set-Cookie" non può essere acceduto

Code Time!

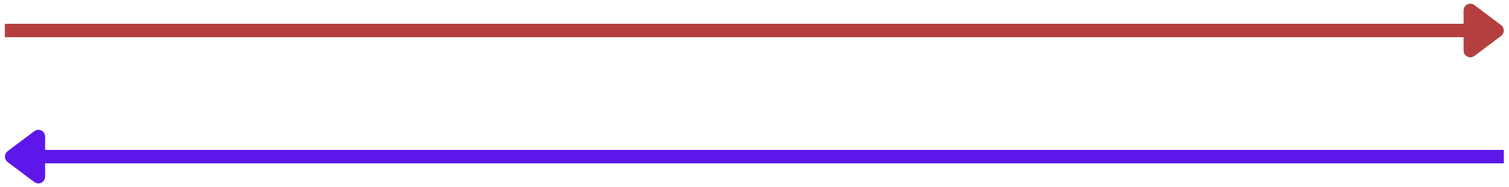
With Credentials:
qual è la parola d'ordine?

Access-Control-Allow-Credentials /1

E' l'intestazione che consente di specificare cookie (o altri tipi di credenziali) nelle richieste "cross origin".



food.acme.spa



accounts.acme.spa

```
GET /account/profile HTTP/1.1
Host: accounts.acme.spa
Cookie: session=bd31d4c2c6d0665f7e8380fa3
Origin: food.acme.spa
...
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: food.acme.spa
Access-Control-Allow-Credentials: true
...
```

Access-Control-Allow-Credentials /2

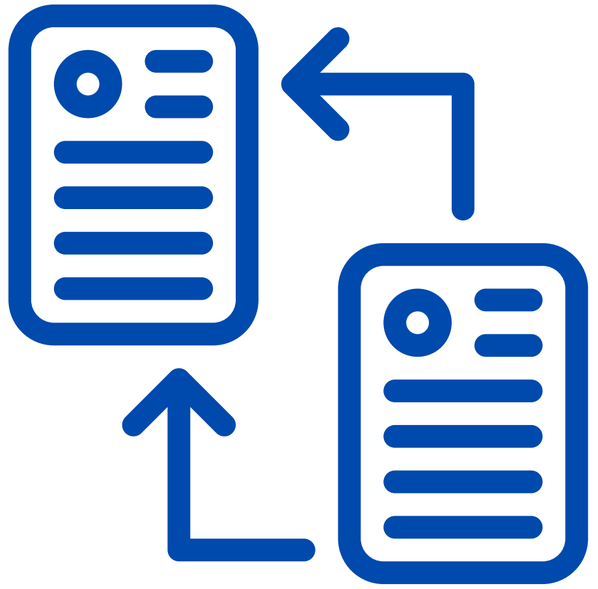
Può essere omesso, oppure indicato così:

```
Access-Control-Allow-Credentials: true
```

Attenzione!

- L'attributo Access-Control-Allow-Origin non può valere "*"
  - Questo renderebbe totalmente inefficace la SOP
  - Chiunque potrebbe accedere anche alle risorse che sono protette da autenticazione

Code Time!



Operazioni più complesse?

Richieste preflight CORS 🛬 /1

Alcune richieste HTTP richiedono una conferma del server per poter essere eseguite, ossia una **richiesta preflight**.

- Metodi diversi da GET, POST(*) o HEAD
- Intestazioni diverse da
  - Accept
  - Accept-Language
  - Content-Language
- (*) Intestazioni Content-Type diverse da
  - multipart/form-data
  - application/x-www-form-urlencoded
  - text/plain

Richieste preflight CORS 🛫 /2

Il browser invia automaticamente una **richiesta OPTIONS** come questa:

```
OPTIONS /api HTTP/1.1
Host: accounts.acme.spa
Origin: food.acme.spa
Access-Control-Request-Method: DELETE
```

Il server deve rispondere con informazioni sulle richieste che può accettare usando le intestazioni

- Access-Control-Allow-Origin
- Access-Control-Allow-Headers
- Access-Control-Allow-Methods
- Access-Control-Max-Age (*)

(*) facoltativa

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: food.acme.spa
Access-Control-Allow-Headers: Content-Type
Access-Control-Allow-Methods:
↩️  GET, DELETE, HEAD, OPTIONS
```

DMVC Framework CORS Middleware

Per configurare CORS in modo dettagliato, è possibile utilizzare i parametri del costruttore del middleware:

```
constructor Create(
  const AAllowedOriginURL: string = TMVCCORSDefaults.ALLOWS_ORIGIN_URL;
  const AAllowsCredentials: Boolean = TMVCCORSDefaults.ALLOWS_CREDENTIALS;
  const AExposeHeaders: String = TMVCCORSDefaults.EXPOSE_HEADERS;
  const AAllowsHeaders: String = TMVCCORSDefaults.ALLOWS_HEADERS;
  const AAllowsMethods: string = TMVCCORSDefaults.ALLOWS_METHODS;
  const AAccessControlMaxAge: Integer = TMVCCORSDefaults.ACCESS_CONTROL_MAX_AGE
); virtual;
```

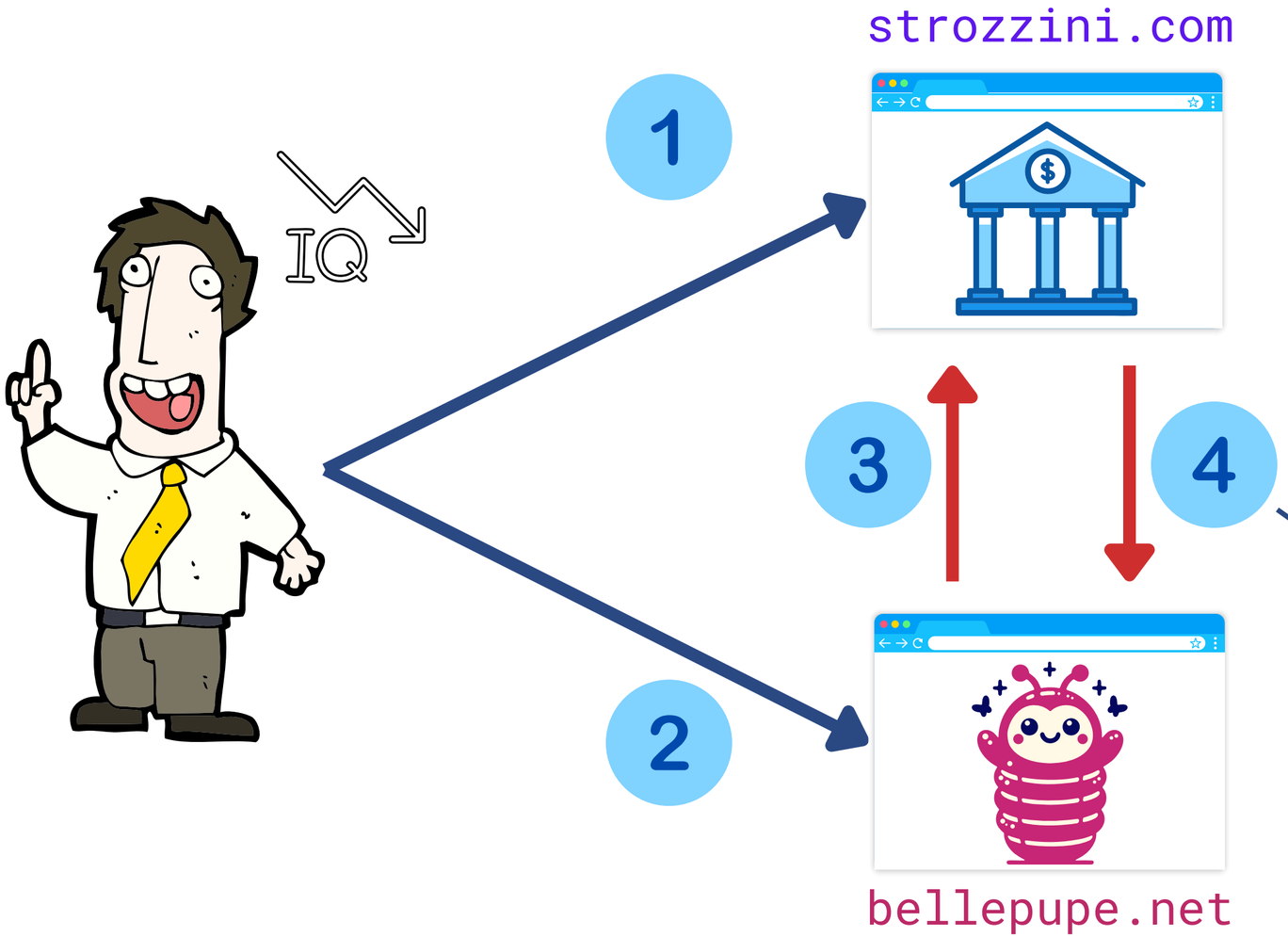
Occhio ai valori di default! 🫡

Code Time!

Problemi di CORS?

...ovvero “Mi sento vulnerabile” 🤫

CORS è vulnerabile! ☠️




```
HTTP/1.1 200 OK
Host: strozzini.com
Access-Control-Allow-Origin: bellepupe.net
Access-Control-Allow-Credentials: true
...
{
  "userName": "terrapiatta64",
  "iban": "IT40N0300203280446469362488",
  "otherVeryPrivateInfo": "..."
}
```

Perché accade questo?

La maggior parte delle vulnerabilità spesso derivano da

- una **errata configurazione**
- **limitate opzioni** per l'header "Access-Control-Allow-Origin"
- **forzatura** degli sviluppatori all'uso della **generazione dinamica** degli header

Come “spezzare” il CORS… 💔

- Valore di Access-Control-Allow-Origin generato prendendo direttamente il valore di Origin! 😱

- Errore nel parsing dell’header Origin dalla richiesta
  - Accesso a tutti i domini che iniziano/finiscono con…
    - Esempio: “store.com”
    - Bypass con “illegalstore.com”

- Inserimento in whitelist del valore “null”

Quali sono gli impatti?

- Variano a seconda dell'applicazione che viene "attaccata", dalle dipendenze e da come è stata implementata
  - Riservatezza (Confidentiality)
    - Nessuna / Parziale (bassa) / Alta
  - Integrità (Integrity)
    - Parziale o Alta
  - Disponibilità (Availability)
    - Nessuna / Parziale (bassa) / Alta

- Nei casi più gravi, l'esecuzione di codice remoto

Come identificare le vulnerabilità? /1

- Può richiedere personale esperto in sicurezza
  - Sono necessarie tecniche per tentativi di exploit

- Black-Box Testing
  - L'header ACAO riflette quello inviato come Origin?
  - Viene validata solo l'inizio o la fine della stringa?
  - Consente l'uso dell'origine "null"?
  - Consente il cambio di protocollo? (es. da HTTPS a HTTP)

- Individuate le possibili vulnerabilità, viene eseguito il test e si richiede il fix dell'applicazione in base agli "issue" individuati

Come identificare le vulnerabilità? /2

- White-Box Testing
  - Identificare gli strumenti utilizzati per l'implementazione
    - Piattaforma e/o runtime
    - Librerie e framework
  - Capire come gli strumenti permettono di configurare nello specifico CORS
    - Quali classi? Quali metodi? Quali parametri?
    - Come funziona il modello a oggetti e/o SDK/API?
  - Fare "code review" per identificare qualsiasi parte errata o pericolosa (code smell) che configura CORS

Attenzione alla cache!

- Le richieste con CORS “subiscono” anch’esse la cache

- Se si usa “*” come Access-Control-Allow-Origin
  - Se non si cambia l’URL della risorsa
  - Se si cambia la policy successivamente
  - Il browser non la vede aggiornata (pesca da cache!)

- Ricordarsi di usare gli attributi “VaryBy” sulla cache
  - In presenza di questi header, anche “*” è safe

Best practices!

- Ricordati innanzitutto di configurare CORS!

- Definisci gli elenchi di accesso appropriati
  - Usa valori di host precisi e ben determinati
  - Assolutamente no regex, no caratteri jolly, no mask

- Evita l'uso dell'origine "null"
  - Pericolosa! Permette chiamate da file HTML!

- Verifica le vulnerabilità di runtime e librerie (es. OWASP)

Risorse e link

Per approfondire l'argomento...

Link e approfondimenti

Cross-Origin Resource Sharing (CORS) - Mozilla Developer Network
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

3 Ways to Fix the CORS Error — and How the Access-Control-Allow-Origin Header Works
https://medium.com/@dtkatz/3-ways-to-fix-the-cors-error-and-how-access-control-allow-origin-works-d97d55946d9

JetBrains IDE Remote Code Execution and Local File Disclosure
http://blog.saynotolinux.com/blog/2016/08/15/jetbrains-ide-remote-code-execution-and-local-file-disclosure- vulnerability-analysis/

Exploiting CORS misconfigurations for Bitcoins and bounties
https://portswigger.net/web-security/cors

Advanced CORS Exploitation Techniques
https://www.corben.io/advanced-cors-techniques/


Q&A

Thank you