A

# La richiesta HTTP spiegata in modo semplice

Nel web i client (in questo caso il browser da voi utilizzato) utilizzano il protocollo HTTP per comunicare con i server web. Questo serve a regolare come i client formulano le proprie richieste e di conseguenza come deve rispondere il server. Il protocollo HTTP contiene diversi metodi di richiesta. Per questo motivo, vale la pena conoscere almeno quelli più comunemente utilizzati. In questo articolo vi spieghiamo come funziona ognuno di questi tipi di richieste HTTP (HTTP request), utilizzando dei semplici esempi.

**Indice**



## GET

GET è l'antenato delle richieste HTTP. Questo metodo di richiesta esiste dagli inizi del World Wide Web e svolge la funzione di **richiedere una risorsa**, come ad esempio un file HTML, **a un web server**.

Digitando l'URL *www.example.com* nella barra di ricerca del vostro browser, questi si mette in collegamento con il server web e gli invia una richiesta di tipo GET:

```
GET /index.php
```

Il file *index.php* riportato in questo esempio non è altro che la pagina iniziale del sito, che verrà inviata dal server al vostro browser in risposta alla richiesta HTTP.

Analogamente, la richiesta dell'URL *www.example.com/test.html* verrebbe formulata così:

```
GET /test.html
```

Così il server invierebbe in risposta il file *test.html*.

### Parametro URL

Alla richiesta GET possono essere allegate **informazioni aggiuntive** da far elaborare al server web. Questi parametri URL vengono semplicemente agganciati all'URL. La sintassi è molto semplice:

- La stringa della query (sinonimo di richiesta) si apre con un "?" (punto di domanda).
- A ogni parametro viene dato un nome, composto da un nome e un valore: "Nome=Valore"
- Nel caso in cui vengano assegnati più parametri, questi vengono collegati tra loro utilizzando il carattere "&".

Esempio: mentre si sta visitando il sito web di un'azienda che sviluppa software si clicca su "Windows" e dunque sulla categoria "Office", per vedere le relative offerte, richiamandole così dal server:

```
GET /search?platform=Windows&category=office
```

### La codifica URL della stringa query

Le stringhe query seguono una codifica particolare, in quanto molti **caratteri speciali** posseggono un significato specifico. Per esempio, il testo "che cos'è HTTP" viene codificato come qui di seguito riportato, per far sì che la stringa query venga accettata:

```
GET /search?thema=che%20cos%E2%80%99%C3%A8%20HTTP
```

Consiglio
Niente paura, non dovete imparare a memoria come si codificano i singoli caratteri speciali. Potete comodamente affidarvi al tool online URL Encoder, o utilizzare la funzione di Excel ENCODEURL (non da web app).

## POST

Nel caso in cui vogliate inviare grandi quantità di dati, come un'immagine o i dati della compilazione di un modulo a un server, il metodo GET non è ideale, in quanto tutti i dati trasmessi vengono riportati apertamente nella barra degli indirizzi del browser. Quello che fa per voi è il metodo POST. Il metodo POST scrive i parametri URL non all'interno dell'URL stesso, ma allegandoli all'header HTTP.

Le richieste POST vengono principalmente utilizzate in relazione ai **moduli da compilare online**. Qui di seguito vi riportiamo l'esempio di un formulario, nel quale viene richiesto il nome e l'indirizzo e-mail, e che viene inviato al server tramite POST:

```
<html>
<body>
<form action="newsletter.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

Consiglio
Nella nostra Digital Guide trovate un articolo di approfondimento sulle differenze tra GET e POST.

## HEAD

Il metodo HEAD serve a richiedere l'header di risposta, o Response Header, senza però che il file venga inviato immediatamente in risposta. Questo torna particolarmente utile e sensato, nel caso in cui a essere trasmessi siano file di grandi dimensioni**.** Grazie alla richiesta HEAD il client riceve per prima cosa l'**informazione sulle dimensioni del file**, così da poter decidere se ricevere il file oppure no.

Esempio:

```
HEAD /downloads/video1.mpeg HTTP/1.0
```

Nell'header di risposta del server, il client trova indicate le dimensioni del file nel campo dell'header "content length":

| | |
|---|---|
| age | 96529 |
| cache-control | max-age=604800 |
| content-length | 1495 |
| content-type | text/html; charset=UTF-8 |
| date | Fri, 06 Mar 2020 14:53:39 GMT |
| etag | "3147526947+gzip" |
| expires | Fri, 13 Mar 2020 14:53:39 GMT |
| last-modified | Thu, 17 Oct 2019 07:18:26 GMT |
| server | ECS (dcb/7F83) |
| vary | Accept-Encoding |
| x-cache | HIT |

Il server risponde alla richiesta HEAD con dati di riferimento sul file richiesto.

## OPTIONS

Con il metodo OPTIONS il client può **richiedere quali metodi supporta il server** per il file in questione.

```
OPTIONS /download.php
```

Un esempio di risposta potrebbe apparire così:

| | |
|---|---|
| allow | OPTIONS, GET, HEAD, POST |
| cache-control | max-age=604800 |
| content-length | 0 |
| content-type | text/html; charset=UTF-8 |
| date | Fri, 06 Mar 2020 14:06:08 GMT |
| expires | Fri, 13 Mar 2020 14:06:08 GMT |
| server | EOS (vny/0452) |

Risposta del server a una richiesta OPTIONS.

Nel campo "allow" scopriamo che il server supporta i metodi OPTIONS, GET, HEAD e POST. Il numero 0 nel campo "content length", ci informa invece che non è stato trasmesso alcun file, ma solamente l'header.

## TRACE

Con il metodo TRACE può essere ricostruito il **percorso di una richiesta HTTP** fino al server e di ritorno dal server al client, ossia il browser in utilizzo. Questo lo potete fare utilizzando il comando *tracert*, ammesso però che utilizziate Windows come sistema operativo. Il funzionamento è semplice. Digitate il seguente comando nella riga di comando (*cmd.exe*):

```
tracert www.example.com
```

## Metodi speciali

Alcuni metodi possono essere utilizzati esclusivamente con determinate configurazioni specifiche. Tra questi vi sono il metodo CONNECT, che stabilisce una connessione diretta e protetta attraverso un proxy (Tunneling), così come anche diversi metodi legati al WebDAV: PATCH, PRPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK.

PUT, PATCH e DELETE servono rispettivamente a **salvare file sul server, a modificarli e a cancellarli**. Nella normale programmazione dei siti web questi metodi non svolgono alcun ruolo chiave, in quanto vengono bloccati dai server per motivi di sicurezza. Vengono però utilizzati nell'ambito del WebDAV e in relazione con le REST API.