

# Présentation d'HTTP pour le développement Web

L'HyperText Transfer Protocol (HTTP) — littéralement « protocole de transfert hypertexte » — est un protocole de communication client-serveur développé pour le Web.

On peut par exemple l'utiliser avec les logiciels Chrome et Apache.

## Vocabulaire

Pour qu'un serveur Web fournisse un service, il utilise un ou plusieurs enregistrement en base de données. On parle alors de **ressources**.

Chaque ressource est identifiable par une **URI** (ou **URL**).

Lorsque l'on obtient un document HTML, XML ou JSON de la ressource, on parle alors de **représentation** de cette ressource.



## Principaux verbes

Le protocole HTTP dispose de 2 verbes principaux : `GET` et `POST`.

### Quand doit-on employer `GET` et `POST` ?

L'utilisation de `GET` est "sûre", c'est-à-dire que l'état de la ressource ne doit pas être modifiée par un `GET`. Il faut donc utiliser `GET` pour des opérations qui ressemblent à des questions ou à des lectures de l'état de la ressource. Exemple : obtention d'un article sur un blog. Par défaut, le verbe HTTP d'un lien HTML est `GET`. Par conséquent, lorsqu'une application Web est bien réalisée, on peut cliquer 1000 fois sur un lien sans que cela n'entraîne de modification sur la base de données.

En revanche, le verbe `POST` est "dangereux", pouvant entraîner une altération de la base de données. Il faut utiliser `POST` quand la requête exprime une création de ressource. Exemple : création d'un article sur un blog.

### Relation avec le HTML

Ainsi, lorsque l'on veut obtenir la représentation d'une ressource (avec le verbe `GET`), on utilise un lien (balise HTML `a`). Et lorsque l'on veut créer une ressource (avec le verbe `POST`), on utilise un formulaire (balise HTML `form`).

# Communications avec HTTP

## Du client vers le serveur

Parfois, le client a besoin de transmettre des paramètres au serveur. Il existe 2 façons de le faire :

- soit par l'URI (exemple : "<http://exemple.com/ressource.php?param1=foo&para2=bar>" (<http://exemple.com/ressource.php?param1=foo&para2=bar>")),
- soit en soumettant un formulaire.

## Du serveur vers le client

Après chaque requête, le serveur répond toujours au client. Sa réponse se compose de 3 éléments :

- un code à 3 chiffres,
- un en-tête (en anglais, "header"),
- un contenu (optionnel, exemple de formats : HTML, XML, ou encore JSON).

## Code HTTP

Le code HTTP représente un statut. Il donne une valeur sémantique à la réponse du serveur, pour aider le client à traiter l'information.

Liste des principaux codes HTTP :

- `200` : Requête traitée avec succès ("OK").  
Exemple : suite à une requête de lecture d'un article par l'utilisateur, ledit article est retourné par le serveur.
- `201` : Requête traitée avec succès avec création d'un document ("Created").  
Exemple : suite à la soumission d'un formulaire pour créer un article par l'utilisateur, ledit article a bien été ajouté dans la base MySQL.
- `301` : Document déplacé de façon permanente ("Moved Permanently").  
Exemple : depuis la mise à jour du code PHP, l'URI d'un article a changé.
- `401` : Une authentification est nécessaire pour accéder à la ressource ("Unauthorized").  
Exemple : une ressource est disponible seulement pour les membres qui sont connectés.
- `404` : Ressource non trouvée ("Not Found").  
Exemple : l'utilisateur utilise l'URI "<http://google.com/foobarbazqux>" (<http://google.com/foobarbazqux>) et aucune ressource n'est trouvée.
- `500` : Erreur interne du serveur ("Internal Server Error").  
Exemple : lorsqu'il y a un bug avec PHP.