### **1 - Méthodes GET et POST**

#### **Exemple de méthode GET :**

* URL : http://example.com/search?query=produit

Requête :  
sql  
Copier le code  
GET /search?query=produit HTTP/1.1

Host: example.com

* Données échangées : Les données sont envoyées via l'URL après le "?" dans la requête (dans cet exemple, query=produit). Les paramètres sont visibles dans la barre d'adresse du navigateur.

#### **Exemple de méthode POST :**

* URL : http://example.com/form

Requête :  
makefile  
Copier le code  
POST /form HTTP/1.1

Host: example.com

Content-Type: application/x-www-form-urlencoded

Content-Length: 27

username=JohnDoe&password=123

* Données échangées : Les données sont envoyées dans le corps de la requête HTTP. Contrairement à GET, les paramètres ne sont pas visibles dans l'URL.

### **2 - Comparaison des méthodes GET et POST**

| **Critère** | **GET** | **POST** |
| --- | --- | --- |
| Visibilité | Paramètres visibles dans l'URL | Paramètres cachés dans le corps |
| Taille des données | Limité par la longueur de l'URL | Pas de limite stricte |
| Sécurité | Moins sécurisé (données visibles) | Plus sécurisé (données dans le corps) |
| Cache | Peut être mis en cache | Ne peut pas être mis en cache |
| Idempotence | Oui (la même requête renvoie le même résultat) | Non (peut modifier les données) |
| Utilisation | Lecture de données | Envoi de données sensibles (formulaires) |

### **3 - HTTP est un protocole extensible**

HTTP est extensible car il permet d'ajouter de nouvelles fonctionnalités grâce à des **extensions** (ex : WebSocket pour la communication en temps réel) et des **en-têtes personnalisés** qui permettent aux clients et serveurs d'ajouter des métadonnées spécifiques pour répondre à des besoins particuliers sans modifier le protocole de base. L'ajout de nouveaux codes de statut, méthodes HTTP (ex : PUT, PATCH), et autres extensions permet d'adapter HTTP à de nouvelles situations.

### **4 - HTTP est un protocole sans état (stateless)**

HTTP est dit "sans état" car chaque requête est indépendante et ne conserve aucune information sur les requêtes précédentes. Cela signifie que le serveur ne se souvient pas des requêtes antérieures, à moins que des mécanismes comme les **cookies**, **sessions** ou **tokens** soient utilisés pour garder une trace de l'état entre les requêtes.

#### **Conséquences :**

* **Avantage** : Évolutivité, car chaque requête est traitée indépendamment.
* **Inconvénient** : Nécessite des mécanismes supplémentaires (cookies, sessions) pour maintenir un état entre les interactions (comme l'authentification ou le suivi des utilisateurs).

### **5 - Décomposition d'une URL**

Exemple d'URL : https://www.example.com:443/search?q=example#section1

| **Partie** | **Rôle** |
| --- | --- |
| https:// | **Schéma** : Indique le protocole à utiliser (HTTP ou HTTPS). |
| www.example.com | **Nom de domaine** : Adresse du serveur hébergeant la ressource. |
| :443 | **Port** : Port utilisé pour la communication (443 pour HTTPS). |
| /search | **Chemin** : Indique la ressource spécifique sur le serveur. |
| ?q=example | **Paramètres de requête** : Informations supplémentaires passées via l'URL. |
| #section1 | **Fragment** : Indique une section spécifique d'une page Web. |

### **6 - Familles de codes statut HTTP**

| **Famille** | **Description** | **Exemple de code** |
| --- | --- | --- |
| 1xx | Informations (La requête est en cours) | 100 Continue |
| 2xx | Succès (La requête a réussi) | 200 OK |
| 3xx | Redirection (Une autre action est requise) | 301 Moved Permanently |
| 4xx | Erreur du client | 404 Not Found |
| 5xx | Erreur du serveur | 500 Internal Server Error |

### **7 - Négociation de contenu HTTP**

Le principe de la **négociation de contenu** permet au client (navigateur) et au serveur de déterminer quel format de réponse doit être utilisé. Cela se fait principalement via les **en-têtes** Accept, Accept-Language et Accept-Encoding envoyés par le client dans la requête. Le serveur sélectionne alors le contenu le plus approprié en fonction des capacités et préférences du client.

#### **Exemple :**

Requête du client :  
bash  
Copier le code  
GET /index.html HTTP/1.1

Accept: text/html,application/xml;q=0.9,\*/\*;q=0.8

* Le serveur répondra avec le contenu text/html si disponible, ou application/xml si le HTML n'est pas disponible, en fonction des préférences du client.

### **8 - Installation Apache & configuration**

Pour installer XAMPP et configurer un VirtualHost :

1. Télécharger et installer XAMPP depuis le site officiel.

Configurer un VirtualHost en éditant le fichier httpd-vhosts.conf dans le répertoire Apache :  
apache  
Copier le code  
<VirtualHost \*:80>

DocumentRoot "C:/xampp/htdocs/dev.local"

ServerName dev.local

<Directory "C:/xampp/htdocs/dev.local">

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

Modifier le fichier hosts (sous Windows : C:\Windows\System32\drivers\etc\hosts) pour ajouter :  
lua  
Copier le code  
127.0.0.1 dev.local

### **9 - CURL**

**Requête GET vers http://dev.local** :  
bash  
Copier le code  
curl http://dev.local

* + Résultat : Contenu de la page d'accueil de dev.local.

**Afficher l'en-tête de la réponse** :  
bash  
Copier le code  
curl -I http://dev.local

Résultat :  
css  
Copier le code  
HTTP/1.1 200 OK

Content-Type: text/html; charset=UTF-8

**Requête GET vers une page non existante** :  
bash  
Copier le code  
curl http://dev.local/notExisting

* + Résultat : Une erreur 404.

**Afficher l'en-tête de la réponse pour une page non existante** :  
bash  
Copier le code  
curl -I http://dev.local/notExisting

Résultat :  
mathematica  
Copier le code  
HTTP/1.1 404 Not Found

1. **Téléchargement de fichier** :
   * Uploader un fichier dans le dossier dev.local/download.

Utiliser curl pour le télécharger :  
bash  
Copier le code  
curl http://dev.local/download/file.txt -o file.txt

### **10 - Principaux en-têtes HTTP**

| **En-tête** | **Rôle** | **Exemple** |
| --- | --- | --- |
| Host | Indique le serveur cible (nécessaire pour les serveurs virtuels). | Host: dev.local |
| User-Agent | Indique le client (navigateur) effectuant la requête. | User-Agent: Mozilla/5.0 |
| Accept | Indique les types de contenu acceptés par le client. | Accept: text/html,application/json |
| Content-Type | Indique le type de données envoyées (utilisé avec POST). | Content-Type: application/x-www-form-urlencoded |
| Authorization | Contient les informations d'identification pour authentification. | Authorization: Basic YWxhZGRpbjpvcGVuc2VzYW1l |
| Cache-Control | Gère la mise en cache de la réponse. | Cache-Control: no-cache |