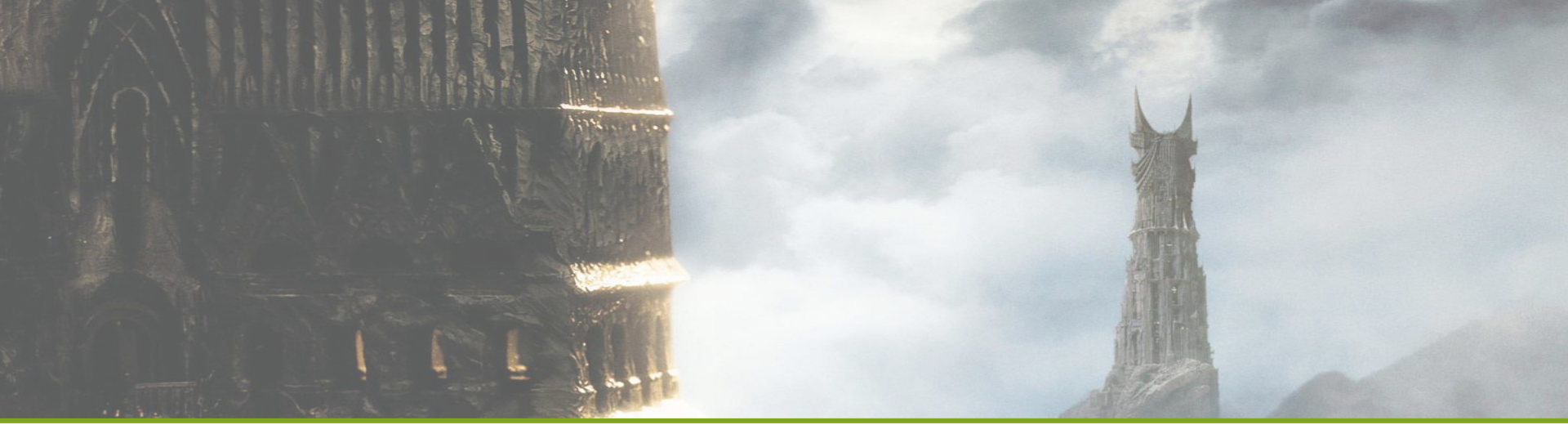
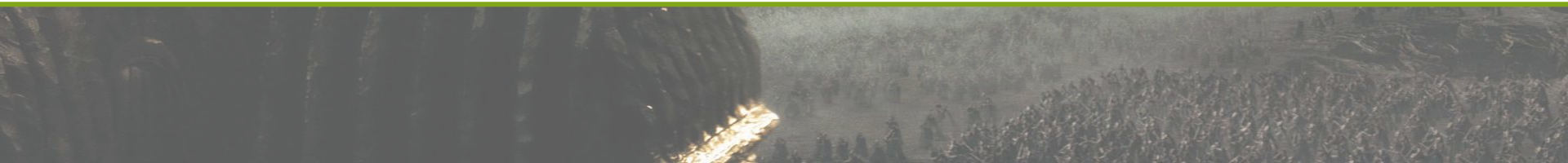


HTTP Fondamentaux

ESGI - 2016 / 2017



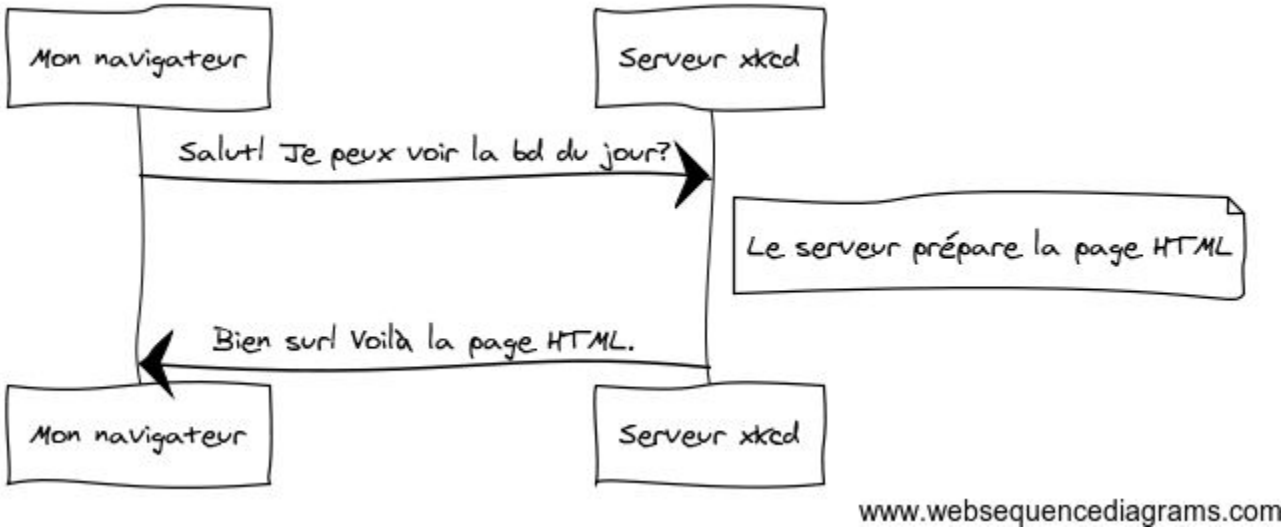
Plan

Http is simple but efficient.

- Requête
- Réponse
- Framework
- Architecture REST
- HTTP/2

Http is simple

Il permet à deux machines de communiquer et … c'est tout !



Le Client envoie une Requête

1. GET / HTTP/1.1
2. Host: xkcd.com
3. Accept: text/html
4. User-Agent: Mozilla/5.0 (Macintosh)

Les verbes HTTP

Définissent les divers moyens par lesquels vous pouvez agir sur la ressource

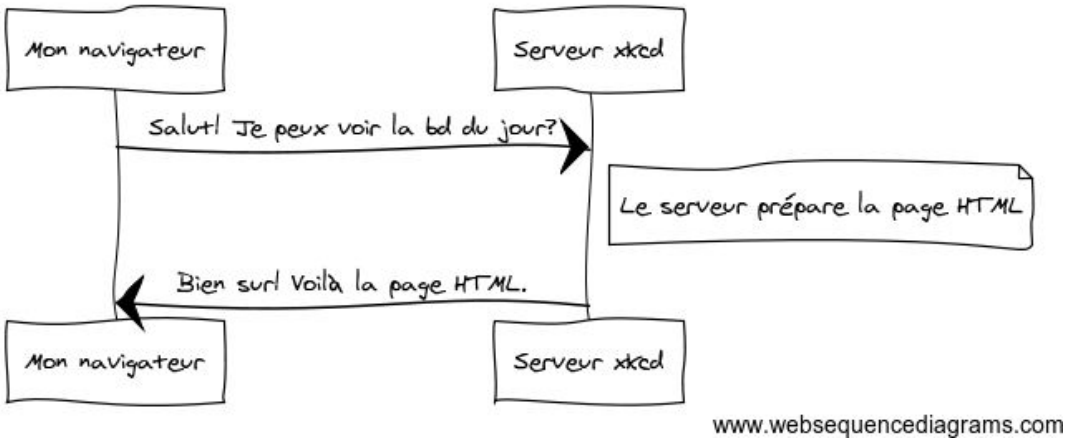
GET	Récupère la ressource depuis le serveur
POST	Crée une ressource sur le serveur
PUT	Met à jour la ressource sur le serveur
DELETE	Supprime la ressource sur le serveur

Des requêtes HTTP

- Demander un compte client
- Mettre à jour les infos sur un client
- Supprimer l’article 15 d'un blog

Le Serveur retourne une réponse

1. HTTP/1.1 200 OK
2. Date: Sat, 02 Apr 2011 21:05:05 GMT
3. Server: lighttpd/1.4.19
4. Content-Type: text/html
5.
6. <html>
7. <!-- ... code HTML -->
8. 8 </html>



Requêtes, Réponses et Développement Web

Cette conversation requête-réponse est le procédé fondamental qui régit toute communication sur le web.

Ainsi le protocole définit les données échangées et la façon de le faire.

Le plus important est que : quel que soit le langage que vous utilisez, le type d'application que vous construisez (web, mobile, API JSON), ou la philosophie de développement que vous suivez, l'objectif final d'une application est toujours de comprendre chaque requête et de créer et retourner la réponse appropriée.

Requêtes et réponses en PHP

PHP vous abstrait une partie du processus global :

```
1. $uri = $_SERVER['REQUEST_URI'];
2. $foo = $_GET['foo'];
3.
4. header('Content-type: text/html');
5. echo 'L\'URI demandée est: '.$uri;
6. echo 'La valeur du paramètre "foo" est: '.$foo;
```

Requêtes et réponses en PHP

La fonction header() crée des entêtes de réponse :

1. HTTP/1.1 200 OK
2. Date: Sat, 03 Apr 2011 02:14:33 GMT
3. Server: Apache/2.2.17 (Unix)
4. Content-Type: text/html
5.
6. L'URI demandée est: /testing?foo=darkmira
7. La valeur du paramètre "foo" est: darkmira

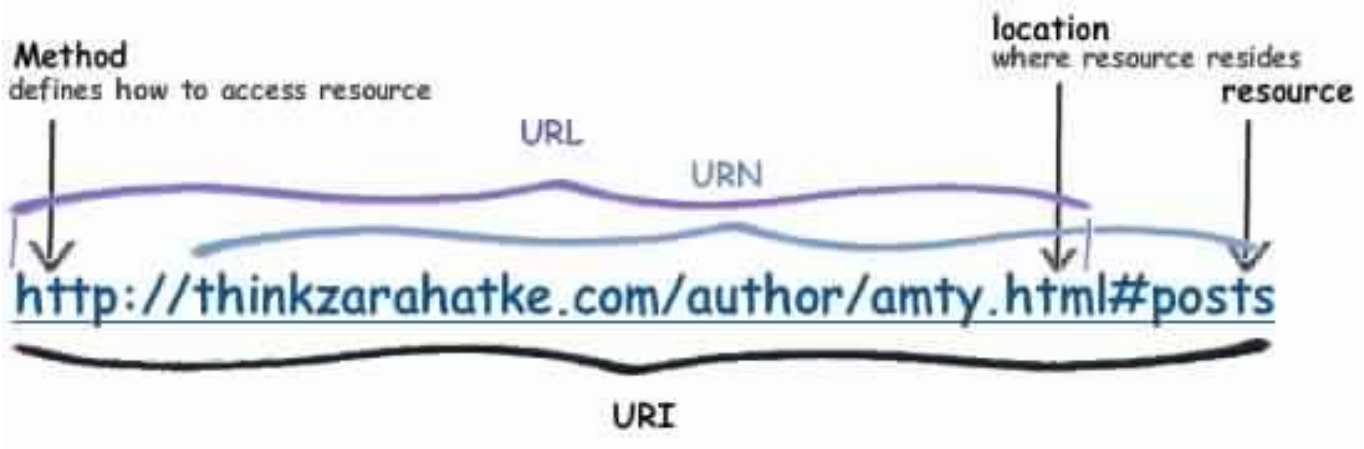
Les frameworks

Finalement ils offrent, et pas uniquement, une boîte à outils pour accéder facilement aux informations de la requête et une interface orientée objet pour créer la réponse.

Comme HTTP lui-même, les objets à disposition sont assez simples. La partie difficile de la création d'une application est d'écrire ce qui vient entre les deux. En d'autres termes, le réel travail commence lors de l'écriture du code qui interprète l'information de la requête et crée la réponse.

URL vs URI

Une image vaut mieux qu'un long discours...



Architecture REST

Les principes de REST ont été théorisés par Roy Fielding :

- Séparation claire entre Client et Serveur
- Le client contient la logique métier, le serveur est sans Etat
- Les réponses du serveur peuvent ou non être mises en cache
- L'interface doit être simple, bien définie, standardisée
- Le système peut avoir plusieurs couches comme des proxys, systèmes de cache, etc
- Éventuellement, les clients peuvent télécharger du code du serveur qui s'exécutera dans le contexte du client

Méthodes HTTP et REST

Méthode	Rôle	Code retour HTTP
GET URL	Récupération élément	200
GET URL	Récupération collection	201
POST URL	Envoi d'éléments	201
DELETE URL	Effacer élément	200
PUT URL	Modifier un élément	200
PATCH URL	Modification partielle élément	200

Erreurs REST

Code Erreur	Description	Signification
400	Bad Request	Requête mal formée
404	Not Found	Ressource demandée inexistante
401	Unauthorized	Authentification nécessaire pour accéder à la ressource
405	Method Not Allowed	Méthode interdite pour cette ressource
409	Conflict	Par exemple, un PUT qui crée une ressource 2 fois
500	Internal Server Error	Autres erreurs du serveur

HTTP2

- Basé sur SPDY de Google
- Conservation des mêmes API que HTTP
- Requêtes moins coûteuses
- Optimisation des connexions TCP avec les serveurs
- Pré-remplissage du cache
- Interruption d'une requête HTTP2 sans la fermer
- Chiffrement des échanges facilité
- La plupart des navigateurs ont soutenu le projet et supporte la nouvelle version du protocole depuis fin 2015

Historique

- 20 ans d’existence
- beaucoup plus que sécuriser les échanges

Basé sur SPDY de Google

- Google a mené ses propres travaux
- réduire le temps de chargement des pages web
  - priorisant les contenus
  - multiplexant leur transfert
- une seule connexion TCP
- L'approche a été reprise par l'IETF

Conservation des mêmes API que HTTP

- La compatibilité ascendante est primordiale
- Les mêmes méthodes (GET, PUT, POST, etc.)
- Les mêmes entêtes et les mêmes statuts
- Les mêmes codes d'erreur (le fameux 404 et tous les autres)

Requêtes moins coûteuses

- HTTP est un protocole coûteux
- notamment car les entêtes de messages sont verbeux
- handicapant pour les informations courtes
- mécanisme de « multiplexage » pour des messages en parallèle et non plus séquentiels
- compression systématique des entêtes

Optimisation des connexions TCP avec les serveurs

- Plus de connexions multiples
  - parallèles
  - simultanées
- une seule et même connexion TCP

Pré-remplissage du cache

- mécanisme de « cache pushing » en une seule fois et pour un usage ultérieur
- possibilité de rendre inactif le mécanisme

Interruption d’une requête HTTP2 sans la fermer

- initier une connexion TCP/IP coûte des ressources
- avec HTTP pour préserver la bande passante, il est nécessaire de la fermer
- HTTP/2 peut maintenir la connexion active à moindre frais

Chiffrement des échanges facilité

- le chiffrement coûte des ressources
- rendre le chiffrement des communications plus performant
- ATTENTION : cela ne garantit pas une meilleure sécurité

HTTP/2 en conclusion

- des améliorations attendues
- rien de magique
- attention à la mise en place, qui pourrait aller à l'encontre du but recherché

- HTTP/3 ?

HTTP vs Websocket

- Une limite de HTTP, échange unidirectionnel
- AJAX a permis une certaine interactivité

Websocket pour des communications full-duplex

- L'upgrade est demandé par le client, par en handshake en HTTP
- Si le serveur l'accepte, le nouveau protocole sera websocket

HTTP vs Websocket

Côté client :

```
new WebSocket("ws://127.0.0.1:8004/ws.php")
```

- **onopen** déclenché une fois la connexion établie
- **onmessage** déclenché à la réception d'un message
- **onclose** déclenché une fois la connexion fermée
- **onerror** déclenché lors de l'apparition d'une erreur