Programmation Web coté Serveur

Introduction

Architectures pour le web

Technologies côté client et côté serveur

Protocole HTTP

Introduction : le web

- Le World-Wide Web est un grand système d'information réparti sur un ensemble de sites connectés par le réseau Internet.
- Ce système est essentiellement constitué de documents (ressources) hypertextes, ce terme pouvant être pris au sens large :
  - textes,
  - images,
  - sons,
  - vidéos, etc.

Introduction : le web

- Chaque site propose un ensemble plus ou moins important de documents ou ressources qui sont transmis sur le réseau par l'intermédiaire d'un programme serveur.
- Ce programme serveur dialogue avec un programme client qui peut être situé n'importe où sur le réseau.
- Le programme client prend le plus souvent la forme d'un navigateur, grâce auquel un utilisateur du Web peut demander et de consulter très simplement des documents.

Introduction: Composantes du Web

- Le concept d'hyperlien est au coeur de la conception web
- Pour qu'il fonctionne => 3 composantes:
  - Une méthode universelle pour définir de façon unique une ressource Web => URL (Uniform Resource Locator)
  - Un mécanisme de formatage de document => HTML (Hypertext Markup Language)
  - Un moyen de tout relier => Protocole de communication HTTP (Hypertext Transfer Protocol)

Introduction : le web

- Le dialogue entre un programme serveur et un programme client s'effectue selon des règles précises qui constituent un protocole.
- Le protocole du Web est le HTTP (Hyper Text Transfer Protocol), mais il est souvent possible de communiquer avec un site via d'autres protocoles, comme par exemple le FTP (File Transfer Protocol) qui permet d'échanger des fichiers.

Définitions

- Internet :
  - Réseau des réseaux
  - Fournir une large quantité d'informations pour la communauté scientifique
  - Contrôlé par le World Wide Web Consortium (W3C)
- Intranet
  - Réseau d'entreprise fondé sur les protocoles et les applications d'Internet
  - Web interne, web privé

Comparaison internet/intranet

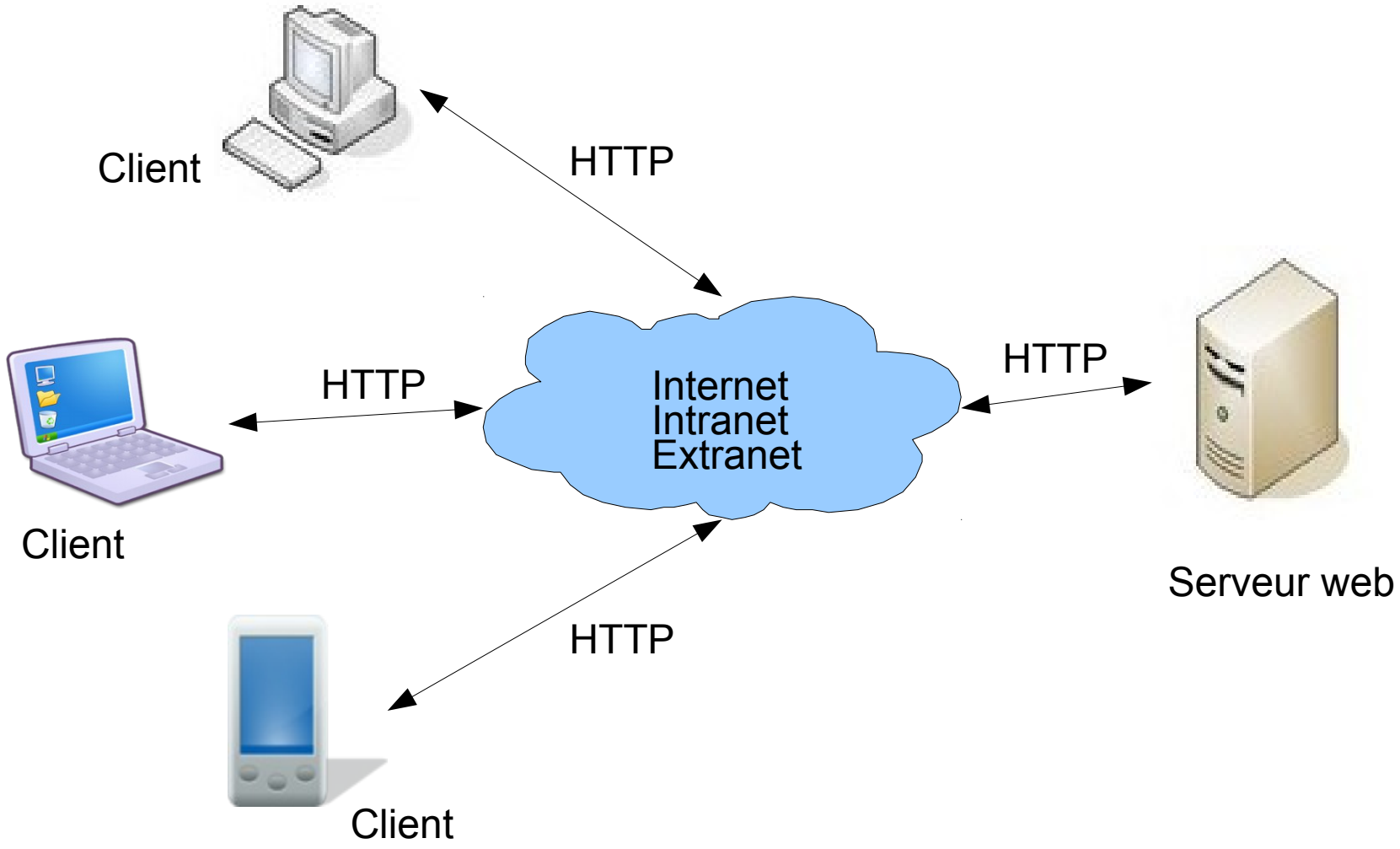
- Information publique
- Protocole IP
- Coût déployé sur les acteurs
- Gestion difficile
- Sécurité faible
- Pas de loi
- Pas de contrôle

- Information privée
- Protocole IP
- Coûts supportés par l'entreprise
- Géré par l'entreprise
- Sécurité forte
- Loi de l'entreprise
- Contrôle d'entreprise

Définitions - suite

- Extranet
  - Applications et services d'un intranet utilisés à l'extérieur de l'entreprise
  - Externalisation d'un intranet
  - Exemple : ENT
- Portail
  - Site web regroupant dans un même espace un ensemble d'informations et de liens cohérents pour un profil d'utilisateur donné
  - Exemples : yahoo, orange, ...

Composants d'une application web



Composants d'une application web : serveur

- Serveur HTTP
  - Frontal d'accès aux services applicatifs
  - Application en attente de connexions des clients sur un port précis
  - Outil de mise à disposition de pages visualisées sur le poste client
  - Frontal de sécurité d'accès au SI
  - Ex : Apache, IIS, Lighttpd, ...

Composants d'une application web : client

- Navigateur :
  - Outil universel de navigation sur le réseau
  - Affiche les informations accessibles par l'utilisateur
  - Client léger
  - Ex : Chrome, Firefox, Safari, Opéra…
- Application utilisateur :
  - Telnet, Skype…
- Araignée (robot pour traverser le web)
- …

Composants d'une application web : réseau

- Le réseau
  - Couche middleware
  - Compatible IP
  - Protocole de transport et de communication : HTTP

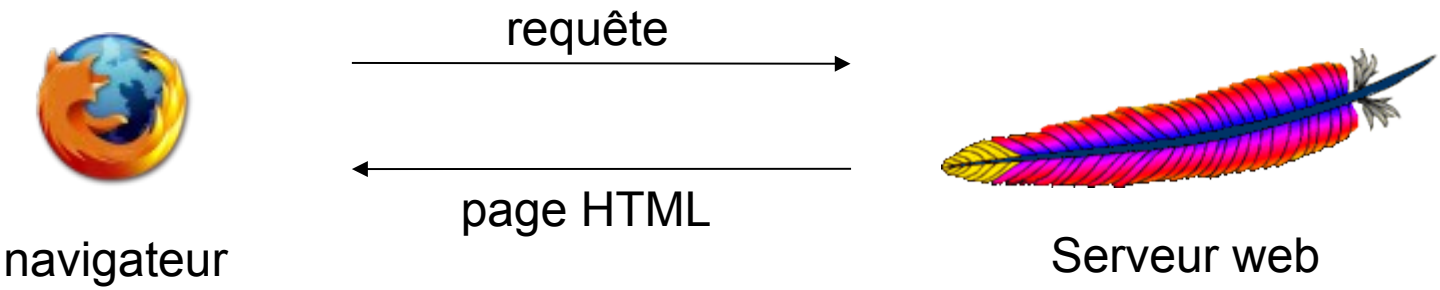
Modèle logique d'architecture applicative

- L'architecture classique utilisée pour les applications web est l'architecture client/serveur. C'est une architecture 2 tiers.
- Chaque tiers a une préoccupation particulière.
- Une évolution est l'architecture 3-tiers où on introduit un tiers pour la persistance des données : on a donc le client, le serveur applicatif et le SGBD.

Communication Web : client - serveur

- Le programme serveur est en attente de **requêtes** transmises à son attention sur le réseau par un programme client.
- Quand une requête est reçue, le programme serveur l'analyse afin de déterminer quel est le document demandé, recherche ce document et le transmet au programme client.
- Il s'agit de communication de type **statique**.
- Dans ce cas, le serveur web peut être vu comme un serveur de fichiers.

Pages web statiques



- Le client émet une requête depuis son navigateur
- Le serveur web (apache par ex) renvoie une page HTML

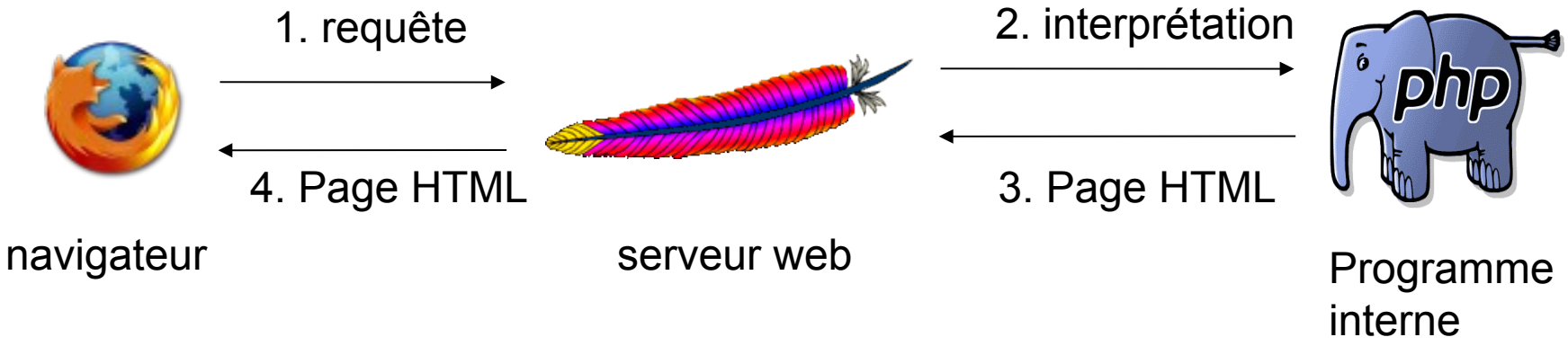
Communication Web : client - serveur

- Le schéma général de communication client-serveur admet beaucoup de variantes :
  - au lieu de demander un document, le programme client peut lui-même transmettre des informations et demander au programme serveur de les conserver. Dans ce cas, le message renvoyé est simplement destiné à informer le programme client que sa demande a été (ou non) satisfaite.
  - un autre type, important, d'interaction consiste pour le programme client à demander au programme serveur d'exécuter un programme, en fonction de paramètres, et de lui transmettre le résultat.
- Il s'agit de communication de type **dynamique**.

Pages dynamiques avec appel à un programme interne

- Le traitement applicatif serveur est mélangé dans la page HTML sous forme de script puis interprété lors de l'appel de la page par le client
- Langages de script :
  - Active Server Page de Microsoft
  - Java Server Page de Sun
  - PHP

Pages web dynamiques avec appel à un programme interne



- Le serveur web interprète le script contenu dans la page HTML avant de la renvoyer au client.

Technologies côté client

- HTML
- CSS
- Javascript
- XML + XSL
- XHTML
- Ajax

Web 2.0

- Notion de collaboration entre utilisateurs
- Services et applications en ligne :
  - Blogs
  - Wiki
  - Flux RSS
  - Réseaux sociaux
  - …
- Support aux échanges entre personnes

Protocole HTTP

- Protocole de niveau application suffisamment léger et rapide, pour la transmission de documents distribués et multimédia à travers un système d'information multi-utilisateurs.
- Protocole générique et sans état pouvant être utilisé pour de nombreuses tâches, dans les serveurs de noms et la gestion d'objets distribués par l'extension de ses méthodes de requêtes, codes des erreurs et entêtes.
- Une caractéristique de HTTP est son typage des représentations de données, permettant la mise en oeuvre de systèmes indépendants des données transférées.

Protocole requête/réponse, sans état

Histoire du protocole

- HTTP est utilisé depuis 1990 (W3 Global Infor. Initiative)
- HTTP/0.9 : première version. Un simple protocole pour le transfert de données brutes à travers Internet.
- HTTP/1.0 : RFC 1954 datée de 1996. Les messages sont au format MIME et contiennent des méta-informations concernant les données transférées et les sémantiques concernant les requêtes-réponses.
- HTTP/1.1 : RFC 2616, 1999. Prise en compte des effets liés aux hiérarchies des proxies, des caches, en plus des connexions persistantes et les hôtes virtuels.
- HTTP/1.1 : juin 2014 : éclatement en 8 pour faciliter l'évolution du protocole HTTP : RFC 7230/7231/7232/7233/7234/7235/7236/7237.
- HTTP/2.0 : en cours : la principale évolution va consister à abandonner le texte comme mode de transport au profit d'un encodage binaire

Fonctionnement global du protocole

- La plupart des communications HTTP sont initiées par le « User Agent » et consistent en une requête devant être appliquée à une ressource sur son serveur origine.
- Dans le cas le plus simple, ceci peut être réalisé par une simple connexion entre l'utilisateur (UA) et le serveur origine (O).
- Une situation plus complexe peut apparaître lorsque un ou plusieurs intermédiaires sont présents dans la chaîne de communication. On trouvera trois types d'intermédiaires: les proxies, les routeurs, et les tunnels (voir la norme pour leur définition).

Fonctionnement global du protocole

- Le protocole HTTP est protocole **requête/réponse**.
- Un client établit une connexion vers un serveur et lui envoie une requête sous la forme d'une **méthode**, d'une **URI**, du **numéro de version**, suivi d'un **message de type MIME** contenant les modificateurs de la requête, les informations sur le client, et éventuellement un **corps**.
- Le serveur répond par une **ligne d'état**, incluant la **version de protocole** et un message de **succès ou d'erreur**, suivi d'un **message** de type MIME contenant l'information sur le serveur, méta-information, et le **corps** éventuel de la réponse.

Les messages HTTP : types de message

- Les messages HTTP consistent en des requêtes émises par un utilisateur à destination d'un serveur, ou d'une réponse d'un serveur au client.
- Syntaxe HTTP :

```
HTTP-message = Request | Response
```

- Les requêtes et les réponses utilisent un format générique (RFC 822) pour le transfert des entités (la charge utile du message)

Les messages HTTP : types de message

- Tous type de message peut contenir une en-tête optionnelle ainsi que le corps de l'entité.
- Le corps et l'en-tête de l'entité sont séparés par une ligne vide (soit une séquence CRLF) :

```
generic-message = start-line
                  *(message-header CRLF)
                   CRLF
                  [ message-body ]

start-line = Request-Line | Status-Line
```

Requête HTTP

- Une requête d'un client vers un serveur inclue, dans sa première ligne, la méthode appliquée à la ressource, l'identificateur de cette ressource, la version de protocole courante et se termine par CRLF. Les éléments sont séparés par les caractères SP. Ni CR ni LF sont admis à l'exception de la séquence finale CRLF.

```
Request = Request-Line
          *(( general-header | request-header
              | entity-header ) CRLF)
          CRLF
          [ message-body ]
```

Requête HTTP

```
Request-Line = Method SP Request-URI  SP
                    HTTP-Version CRLF
Method = "OPTIONS" | "GET" | "HEAD" | "POST" |
          "PUT" | "DELETE" | "TRACE" | "CONNECT” |
          extension-method
```

- La méthode indiquée en tête de ligne est destinée à être appliquée à l'URI cible. Son nom dépend de la casse.
- La ressource donnée par son URI.
- La version HTTP.

Request URI

- L'URI identifiant la ressource réseau à laquelle doit être appliquée la méthode.

```
Request-URI = "*" | absoluteURI | abs_path |
                    authority
```

- Ces 4 options dépendent de la nature de la requête.
- L'astérisque « * » signifie que la requête n'implique pas une ressource particulière mais le serveur lui même. Elle est donc permise avec les méthodes n'impliquant pas nécessairement des ressources.
- L'option « authority » est utilisée avec la méthode CONNECT.

Exemples

- `OPTIONS * HTTP/1.1`
- `GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.0`
- `GET /pub/WWW/TheProject.html HTTP/1.1`

  `Host: www.w3.org`

- Notez que le chemin absolu ne peut être vide; si la ressource se trouve dans la racine (pas de chemin d'accès) le chemin spécifié devra comporter au moins le caractère slash("/") :

  `GET / HTTP/1.1`

La méthode GET

- La méthode `GET` signifie "récupérer" le contenu quel qu'il soit de la ressource (sous forme d'une entité) identifiée par l'URI-visée.
- La sémantique de la méthode `GET` introduit une notion conditionnelle si la requête inclue le champ d'en-tête `If-Modified-Since`.

La méthode POST

- La méthode `POST` est utilisée pour indiquer au serveur de soumettre l'entité contenue dans le message à la ressource identifiée par l'URI-visée.
- `POST` est destinée à fournir un moyen uniforme pour les opérations suivantes :
  - Annotation de ressources existantes;
  - Envoi d'un message vers une édition en ligne, un groupe de nouvelles, une liste d'adresse, ou listes similaires;
  - Fournir un bloc de données, par exemple, un résultat de formulaire, à un processus de gestion de données;
  - Ajout d'éléments à une base de données.

La méthode HEAD

- La méthode `HEAD` est identique à la méthode `GET`, sauf qu'elle ne demande pas la transmission du corps d'entité de la ressource en retour. Seule les métainformations constituant l'en-tête HTTP de la ressource sont envoyées.
- Elle est utilisée à des fins de tests d'hyperliens, vérification d'existence ou d'actualité d'une ressource.
- Il n'existe pas de méthode `HEAD` conditionnelle comme pour la méthode `GET`. Si un champ d'en-tête `If-Modified-Since` est présent dans une requête `HEAD`, il devra être ignoré.

En-têtes de requête HTTP

- Les entêtes de requêtes HTTP sont des méta-données permettant au client de passer des informations supplémentaires sur lui même et sur la requête qu'il envoie.
- Ces entités sont des sortes de paramètres (au sens d'un langage de programmation) que le client passe au serveur.

En-têtes de requête HTTP

```
request-header =

  Accept                  | If-Modified-Since
| Accept-Charset          | If-None-Match
| Accept-Encoding         | If-Range
| Accept-Language         | If-Unmodified-Since
| Authorization           | Max-Forwards
| Expect                  | Proxy-Authorization
| From                    | Range
| Host                    | Referer
| If-Match                | TE
                          | User-Agent
```

Préférences du client

- Les en-têtes indiquant les types de document, encodage de caractères, format de compression et langue du contenu peuvent contenir plusieurs valeurs.
- On peut associer un coefficient de préférence pour chacune de ces valeurs.
- Ce coefficient est un décimal à 3 chiffres maximum après la virgule compris entre 0 et 1 (0 indique une valeur inacceptable).

Exemple

```
GET / HTTP/1.1
Host: www.univ-lr.fr
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac
OS X 10.6; fr; rv:1.9.2.6) Gecko/20100625
Firefox/3.6.6
Accept: text/html,application/xhtml+xml,applica
tion/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;
q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

Réponse HTTP

- Une fois la requête reçue et interprétée, un serveur répond par un message HTTP de réponse :

```
Response = Status-Line
           *(( general-header  | response-header
              | entity-header ) CRLF)
            CRLF
            [ message-body ]
```

Ligne d'état

- La première ligne d'un message de réponse est la ligne d'état, constituée d'une indication du numéro de version du protocole, suivi du code numérique de la réponse, suivi enfin d'une explicitation textuelle de cette réponse, chaque élément étant séparé par un espace. Aucun CR ni LF ne peuvent y apparaître à l'exception de la séquence CRLF finale.

```
Status-Line = HTTP-Version SP Status-Code SP
                        Reason-Phrase CRLF
```

Code d'état et raison

- L'élément code d'état est un nombre entier à 3 chiffres indiquant le succès ou la cause d'erreur de la transaction.
- L'élément "raison" est un commentaire textuel destiné à identifier explicitement la cause d'erreur.
- Le code d'état sera en général exploité par des automates. La raison est à destination de notre intellect humain. Celle-ci n'est pas obligatoirement traitée ni reportée par le client.

Code d'état

- Le premier chiffre du code d'état indique la classe générale de la réponse. Les deux derniers n'ont pas de rôle de classification particulier.
- Le premier chiffre peut prendre 5 valeurs :
  - 1xx Information :La requête est reçue, le processus continue
  - 2xx Succès : L'action a été correctement reçue, interprétée, comprise et acceptée,
  - 3xx Redirection : Une décision supplémentaire doit être prise pour terminer la requête
  - 4xx Erreur Client : La requête présente une erreur de forme ou ne peut être satisfaite
  - 5xx Erreur Serveur : La requête est valide, mais le serveur ne peut la satisfaire

Les en-têtes de réponse HTTP

- Les entêtes de réponses HTTP sont des méta-données permettant au serveur de passer au client des informations supplémentaires concernant lui même et la réponse qui délivre.

```
response-header = Accept-Ranges
                | Age | ETag
                | Location| Proxy-Authenticate
                | Retry-After | Server
                | Vary   | WWW-Authenticate
```

Exemple

```
HTTP/1.1 200 OK\r\n
Server: Apache\r\n
ETag: "9fe870f-2ca-478d2022"\r\n
Accept-Ranges: bytes\r\n
```

Les en-têtes HTTP générales

- Les entêtes HTTP générales s'appliquent sur les requêtes et les réponses. Elles donnent des informations sur le message qui a été transféré (ne pas confondre avec le corps de l'entité HTTP).

```
general-header = Cache-Control
                 | Connection | Date
                 | Pragma | Trailer
                 | Transfer-Encoding
                 | Upgrade | Via
                 | Warning
```

Le corps d'entité

- Le corps d'entité (s'il existe) envoyé dans un message de requête ou de réponse HTTP est dans un format et sous un encodage défini par les champs d'en-tête d'entité.

  ```
  entity-body = *OCTET
  ```

- Le corps d'entité est présent dans un message seulement si le corps du message est aussi présent.
- Le corps de l'entité est obtenu à partir du corps du message en décodant tout encodage de transfert qui peut être appliqué au message envoyé dans un but de sécurité.

Type et encodage du corps de l'entité

- Lorsqu'un corps d'entité est présent dans le message, le type de données incluses dans ce corps est précisé par les champs d'en-tête Content-Type et Content-Encoding. Ceux-ci définissent un modèle d'encodage arborescent à deux niveaux :

```
entity-body := Content-Encoding ( Content-Type
                    ( data ) )
```

Les en-têtes d'entité HTTP

- Les entêtes des entités HTTP sont des informations sur le message transporté par les requêtes et les réponses. Lorsqu'un message HTTP inclut une entité (un corps) il y a des entêtes optionnelles et d'autres obligatoires.

```
entity-header  =
Allow | Content-Encoding | Content-Language
| Content-Length | Content-Location | Content-MD5
| Content-Range  | Content-Type| Expires
| Last-Modified  | extension-header
```