Le protocole HTTP

Michael Mrissa

Université de Pau et des Pays de l’Adour

Remerciements

- Merci à Lionel Médini et Olivier Glück pour leurs supports

Rappel

- HTTP : Hyper Text Transfer Protocol
  - Dédié au Web (origine : CERN, 1990)
  - RFC 2616 (HTTP 1.1)
  - Fonctionne en mode client / serveur
    - Pas de notifications possible de base (mais extensions)
  - Port standard : 80
    - Pourquoi le port 80 est important ?
    - Adoption massive, d’abord pour les sites Web
    - Puis pour les applications

Rappel

- Protocole sans état
  - Attention cependant
    - Les états existent bien côté serveur et client
  - Gestion légère des transactions
    - aucune information conservée entre 2 connexions
    - permet au serveur HTTP de servir plus de clients
      - Pas d’espace mémoire à allouer à chaque client
  - Nécessite un mécanisme de gestion des sessions
    - cookie, Id dans l’URL, champ caché de formulaire...

Historique

- HTTP 0.9 : version d’origine
  - Une seule méthode : GET
  - Pas d'en-têtes
  - Une requête = une connexion TCP
- HTTP 1.0 : améliorations (1)
  - introduction d’en-têtes (échange de "méta" info)
  - utilisation de caches
  - méthodes d'authentification...

Historique

- HTTP 1.1 : améliorations (2)
  - mode connexions persistantes par défaut
    - plusieurs transactions HTTP (ressources) pour une connexion TCP
    - la connexion est maintenue tant que le serveur ou le client ne décident pas de la fermer (connection: close)
  - introduction des serveurs virtuels
    - la directive Host dans la requête est nécessaire

Historique

- HTTP 2.0 : principes
  - Fondé sur le protocole SPDY (Google)
  - RFC 7540 (mai 2015)
  - Conservation de la syntaxe HTTP 1.1 (méthode, codes de statut, headers...)
  - Ajouts
    - Push serveur de ressources nécessaires
    - Multiplexage des requêtes
    - Compression des headers
    - Couche Sécurité (TLS) obligatoire de fait

Format des requêtes

- Commande HTTP
  - Méthode : GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT
  - URL à partir de la racine du serveur
  - Version HTTP
- En-têtes (ensemble de lignes)
  - Nom de l’en-tête
  - Deux-points
  - Valeur de l’en-tête
- Une ligne vide
- Contenu (éventuel)
  - Passage de paramètres à traiter par le serveur

La méthode GET

- Méthode standard de requête d'un document
  - récupérer un fichier, une image...
  - activer un programme en lui transmettant des données
- Le corps (contenu) de la requête est toujours vide
- Ajout de paramètres après le nom de la ressource
  - Transmission des données dans l'URL après un ?
  - Les champs sont séparés par un &
- GET /index.php?email=toto@site.fr&pass=toto&s=login HTTP/1.1
- Remarques
  - Toutes les données sont transmises en clair et visibles dans l’URL
  - L'URL a une taille limitée (4Ko)

La méthode POST

- Transmission des données dans le corps de la requête
- Les données sont également transmises en clair

```
POST /directory/index.php HTTP/1.1
User-Agent: Mozilla/5.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: localhost
Accept: */*
Content-type: application/x-www-form-urlencoded
Content-length: 36

email=toto@site.fr&pass=toto&s=login
```

La méthode HEAD

- Identique à GET
  - Corps de la requête toujours vide
  - Permet de récupérer seulement l'en-tête de la réponse
- Utilité : récupérer
  - date de dernière modification (caches, JavaScript)
  - taille (estimation du temps d'arrivée du document)
  - type (le client peut sélectionner le type de documents qu'il accepte)
  - Récupérer le type du serveur (requêtes spécifiques au type de serveur)
- Remarque
  - Le serveur ne fournit pas nécessairement ces informations

Quelques en-têtes de requêtes

- Identification du client
  - From : adresse mail du client
  - Host : serveur, obligatoire depuis HTTP 1.1
  - Referer : URL d'où l'on vient
  - User-Agent
- Préférences du client
  - Accept : liste des types MIME acceptés
  - Accept-Encoding : compress, gzip...
  - Accept-Langage
  - Accept-Charset

Quelques entêtes de requêtes

- Objectif
  - Ne pas envoyer un objet que le client a déjà dans son cache
- Problème
  - Les objets dans le cache peuvent être obsolètes
- Solution
  - Le client spécifie la date de la copie cachée dans la requête HTTP
  - If-modified-since : date
  - La réponse du serveur est vide si la copie cachée est à jour

Quelques en-têtes de requêtes

- Information pour le serveur
  - Autorization (username:passwd encodé en base64)
  - Cookie
- Conditions sur la réponse
  - If-Modified-Since : utile pour les caches
  - If-Unmodified-Since
  - If-Match (Etag)

Format des réponses

- Type de la réponse
  - Version HTTP
  - Code de la réponse
  - Description du code
- En-têtes (ensemble de lignes)
  - Nom de l’en-tête
  - Deux-points
  - Valeur de l’en-tête
- Une ligne vide
- Contenu éventuel
  - Ressource encodée en fonction du type MIME spécifié

Codes de réponses

- Les codes de réponse
  - Indiquent le résultat de la requête : succès ou échec
  - En cas d'échec, le contenu de la réponse en décrit la raison
  - Ex : fichier non présent, problème de droit
- Classes de codes
  - 100-199 : information
  - 200-299 : succès
  - 300-399 : redirection
  - 400-499 : échec dû au client
  - 500-599 : échec dû au serveur
- Plus d'infos
  - http://www.codeshttp.com/

HTTP/1.1 200 OK
HTTP/1.1 304 Not Modified
HTTP/1.1 403 Forbidden
HTTP/1.1 404 Not Found
HTTP/1.1 500 Internal Server error

Quelques en-têtes de réponses

- Contenu du document
  - Content-Type : type MIME du document
  - Content-Length : barre de progression du chargement
  - Content-Encoding, Content-Location, Content-Langage
- Document lui-même
  - Last-Modified : date de dernière modification
  - Allow : méthodes autorisées pour ce document
  - Expires : date d'expiration du document
- En-tête générales
  - Date : date de la requête
  - Server : type du serveur

Une transaction typique (1)

- Requête du client : client => serveur
  - 1. demande du document test.html

```
GET /~mydirectory/test.html HTTP/1.1
```

  - 2. envoi des informations d'en-tête : informer le serveur
    - configuration
    - documents acceptés

```
User-Agent: Mozilla/5.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: www.univ-pau.fr
Accept: image/gif, image/jpeg
```

  - 3. envoi d'une ligne vide (fin de l'en-tête)
  - 4. envoi du contenu (vide dans cet exemple)

Une transaction typique (2)

- Réponse du serveur : serveur client
  - 5. code indiquant l'état de la requête

    HTTP/1.1 200 OK
  - 6. envoi des informations d'en-tête : informer le client
    - configuration du serveur
    - document demandé

    Date: Tue, 30 Sep 2008 06:11:28 GMT
    Server: Apache/1.3.34 (Debian) PHP/5.2.1
    Last-Modified: Tue, 30 Sep 2008 06:11:14 GMT
    ETag: "600593b3-61-48e1c302"
    Accept-Ranges: bytes
    Content-Length: 97
    Content-Type: text/html; charset=iso-8859-1
- 3. envoi d'une ligne vide (fin de l'en-tête)
- 4. envoi du contenu si la requête a réussi

Cookies

- HTTP : protocole sans état
  - Nécessite un moyen de gérer les sessions => cookies
- Cookie
  - chaîne de caractères url-encodée de 4ko max stockée sur le disque dur du client
  - informations associées à un ensemble d'URL, utilisées lors de toute requête vers l'une de ces URL
- Les cookies permettent de
  - propager un code d'accès : évite une authentification lors de chaque requête
  - identification dans une base de données
  - fournir des éléments statistiques au serveur : compteurs de pages visitées...
- Remarque
  - Ce n’est pas le seul moyen de gérer les sessions

Installation d'un cookie sur le client

- Directive Set-Cookie dans l'en-tête de la réponse HTTP (envoyée lors de la première connexion)

```
Set-Cookie: nom=valeur; expires=date; path=chemin_accès;
domain=nom_domaine; secure
```

  - nom=valeur : contenu du cookie, sans espace, point-virgule et virgule (seul champ obligatoire)
  - expires : devient invalide après la date d'expiration
  - path=/pub : cookie est valable pour toutes les requêtes dont l'URL contient /pub
  - domain : nom de domaine (associé au serveur) pour lequel le cookie est valable
  - secure : le cookie n'est valable que lors d’une connexion sécurisée

Utilisation d'un cookie par le client

- Avant chaque requête, le client vérifie dans sa liste de cookies s'il y en a un qui est associé à cette requête
- Si c'est le cas, le client utilise la directive Cookie dans l'en-tête de la requête HTTP

```
Cookie: nom1=valeur1; nom2=valeur2; ...
```

- Le serveur peut insérer plusieurs directives Set-Cookie
- Dans la première spécification des cookies :
  - un client peut stocker un maximum de 300 cookies
  - un maximum de 20 cookies par domaine est permis
  - la taille d'un cookie est limitée à 4Ko

Transfert par morceaux (HTTP/1.1)

- La réponse peut être envoyée en plusieurs morceaux
  - Le serveur ne peut pas toujours déterminer la longueur totale de la réponse

```
Transfer-Encoding: Chunked
```

- Chaque morceau est constitué d'une ligne :
  - taille du morceau en hexadécimal
  - données
- Après les morceaux, une ligne :
  - 0 (zéro)
  - éventuellement des en-têtes supplémentaires

Encodage des ressources

- Position du problème
  - Un serveur Web peut servir différents types de ressources :
    - texte, pages Web, images, documents, fichiers exécutables...
- Chaque type de ressource est codé de façon différente
- Un client doit connaître le type de ressource pour pouvoir la traiter
  - visualisation dans un navigateur, utilisation d'un plugin, application externe
- Solution (HTTP et Internet en général)
  - MIME : Multi-purpose Internet Mail Extensions
  - Prise en charge de MIME dans HTTP : depuis V1.0

Encodage des ressources

- Types MIME : composition
  - Type général : text, image, audio, video, application...
  - Sous-type : dépend du type général
  - Exemples : image/gif, image/jpeg, application/pdf, application/rtf, text/plain, text/html
  - En perpétuelle évolution
- Types MIME : utilisation
  - Le serveur positionne le header Content-type
    - Ex : Content-Type: text/html; charset=UTF-8
  - Le client associe chaque type MIME à un type de prise en charge

Encodage des caractères

- Rappel : codage des caractères
  - Principe : assignation d'un entier à chaque caractère d'un texte
  - Ne pas confondre : encodage (jeu) de caractères et type (MIME) de fichiers
- Position du problème
  - Différents jeux de caractères
    - ANSI, Europe occidentale, chinois simplifié, etc.
- Différentes normes d'encodage
  - Dépendent en grande partie des OS et de leur paramétrage
    - Exemple : ASCII, Windows-1252, ISO Latin 1, Unicode 8, 16 ou 32...
  - Transmission via le Web multiplateforme
    - Indépendante de l'OS et de la config du serveur ou du client

Encodage des caractères

- L’encodage des caractères utilisés dans une ressource
  - est considéré comme une sous-partie de l’encodage des ressources
    - lié au type MIME de la ressource
    - lié à la langue de la ressource
  - est indiqué dans les headers HTTP de la réponse
    - Content-Language: en, fr
    - Content-Type: text/html; charset=ISO-8859-1

Encodage des paramètres de la requête

- Format : URL-encoded
  - Permet de coder les données dans l'URL (méthode GET)
  - Encodage fait par le client
  - Syntaxe des URL (RFC 2396)
    - début des paramètres : ?
    - séparation entre le nom du champ et sa valeur : =
    - séparateur de champ : &
    - espaces dans la valeur d'un champ : +
    - caractères réservés : ; / ? : @ & = + $,
    - caractères non-alphanumériques remplacés par %xx (xx = code ASCII du caractère en hexadécimal)
  - Exemple
    - nom_champ1=valeur1&nom_champ2=valeur2&...
  - Cas des champs à valeurs multiples (listes de sélection)
    - nom_liste=valeur1&nom_liste=valeur2&...

Spécification de l’encodage côté serveur

- Fonctionnement
  - le type MIME est positionné à partir de l'extension du fichier demandé (/etc/mime.types)
  - l’encodage de la page renvoyée est issu d’une négociation avec le client (mod_negotiation)
  - http://httpd.apache.org/docs/2.0/content-negotiation.html
  - il est possible de définir qu’une partie d’un site aura un encodage particulier à l’aide de directives dans un fichier .htaccess
- Modules
  - Apache : mod_mime
    - http://httpd.apache.org/docs/2.0/mod/mod_mime.html
  - nginx : ngx_http_charset_module
    - http://nginx.org/en/docs/http/ngx_http_charset_module.html

Remarques sur l’encodage

- Dans un navigateur
  - La requête spécifie les types d’encodage pris en charge par le client
    - Accept, Accept-Language, Accept-Charset, Accept-Encoding
- Comme pour d’autres headers HTTP, l’encodage des caractères peut être indiqué dans une ressource HTML
  - <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/>
- Dans la réponse, si un header HTTP et un élément meta sont contradictoires, priorité est donnée au header du serveur...

Programmation côté serveur

- Principe
  - Exécutions sur le serveur
  - Envoi au client d'une ressource composée dynamiquement
- Différentes technologies
  - SSI : server Side Includes
    - Inclusions de contenus dans la page (préprocesseur)
  - Interpréteurs intégrés au serveur HTTP
    - exemples : PHP, Jakarta, ASP...
    - sous Apache : modules additionnels
  - CGI : Common Gateway Interface
    - appel standardisé d'un programme externe
    - scripts « à la CGI » (CGI-like) : mod_perl / Apache

Programmation côté serveur

- Comment structurer le code ?
  - Selon le client, le projet, les équipes, le matériel...
- 3 préoccupations
  - Accès aux données
  - Traitements
  - Mise en forme
- Des préoccupations supplémentaires
  - Authentification, sécurité des données, performance, passage à l'échelle, etc.
- Comment créer les interfaces ?
  - Cf cours REST

Exercice

- Proposez une architecture pour une application Web de gestion de post-it
  - Essayez différentes alternatives
  - Comparez les avantages et inconvénients
  - Quels traitements côté clients ? Côté serveur ?
  - Comment gérer la sécurité ?
  - Comment seront structurés les clients ?
    - Navigateur ou autre ?
  - Optionnel : quel langage choisiriez-vous ?