

Architecture Client-Serveur

Pr. M. AIT HEMAD

ait.hemad.m@gmail.com

IRISI1

Introduction



Internet

- Internet :
 - un réseau de réseaux
 - des protocoles de communication : TCP/IP
 - nombreuses applications : courrier électronique, transfert de fichiers (ftp), messagerie instantanée, peer-to-peer, **World Wide Web**

3

WEB

- le Web (Tim Berners-Lee, 1989) :
 - système d'information réparti en « pages web » = **documents web**
 - basé sur la notion d'**hypertexte** et la notion d'**hyperliens** permettant de naviguer entre les documents web
 - les documents forment un graphe = « toile »
 - des protocoles de communication **HTTP, HTTPS**
 - des adresses pour nommer les documents : **URL**
 - des langages pour créer les documents : **HTML, CSS, JavaScript**
 - des **navigateurs** qui interprètent les document



Tim Berners-Lee

4

WEB

■ Popular Websites



LinkedIn (2003)



PayPal (1999)



Pinterest (2010)



Dropbox (2008)



Google (1998)



Yandex (1997)



Tumblr (2007)



Twittr (2006)



AuctionWeb (Ebay, 1995)



Amazon (1995)



YouTube (2005)



Reddit (2005)



WWW Project (1991)



Altavista (1995)



Yahoo (1994)



Thefacebook (2004)



Flickr (2004)

7

Standards web

■ Définition

- Un standard est un ensemble de spécifications qui définissent la méthode d'utilisation des technologies web
- C'est une recommandation qui doit être respectée de tous afin d'éviter les différences d'interprétation pour un code identique.

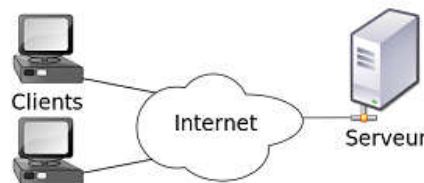
■ W3C "World Wide Web Consortium"

- W3C est l'organisme de normalisation dont le rôle est de créer, de développer, de maintenir et de promouvoir les standards du web.
- Composé en grande partie par plusieurs experts et les éditeurs de navigateurs : Microsoft, Mozilla, Apple, Google...etc.

8

Modèle Client-Serveur

- Modèle de **communication** entre **programmes** à travers un **réseau**.
- Modèle asymétrique:
 - les **Clients** (applications, browsers, programmes ou serveurs) envoient des requêtes,
 - les **Serveurs** (puissance de calcul) traitent les requêtes et répondent.



9

Modèle Client-Serveur

- Caractéristiques d'un serveur
 - Passif
 - À l'écoute (prêt à répondre aux requêtes envoyées par des clients)
 - Réaction (traitement) et réponse aux requêtes
 - Capable de servir plusieurs clients simultanément
- Caractéristiques d'un client
 - Actif
 - Demandeur de services (envoi de requêtes au serveur)
 - En attente des réponses du serveur

10

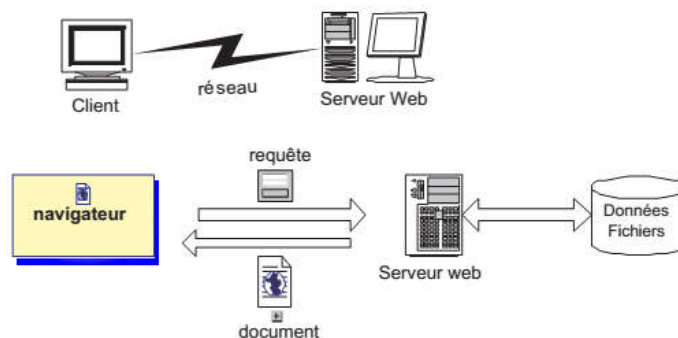
Modèle Client-Serveur

- Avantages:
 - centralisation des données
 - centralisation de la puissance de calcul (clients légers)
- Inconvénients:
 - centralisation des connexions
 - peu robuste

11

Architecture 2-tier

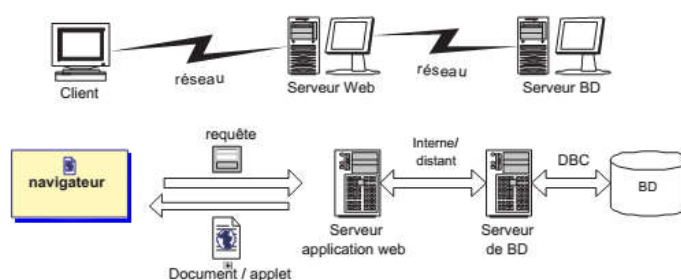
- Client-serveur classique



12

Architecture N-tier

- C'est une extension du modèle client/serveur.
- division entre serveur de données et serveurs d'application
- Exemple : N=3



13

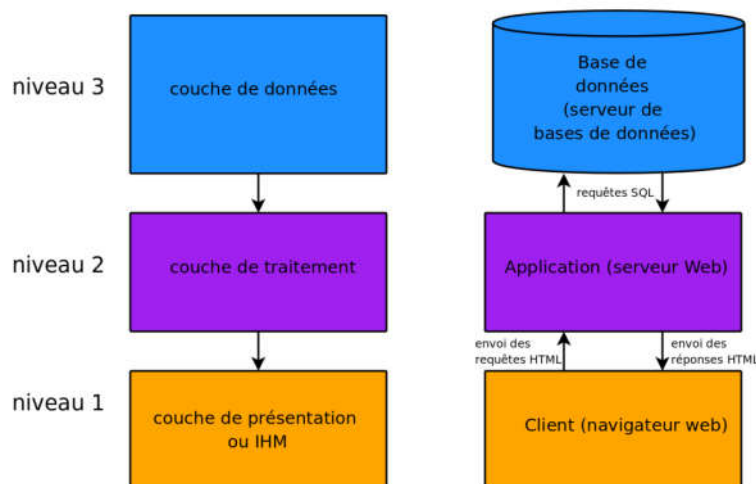
Architecture 3-tier

- L'architecture 3-tier ou architecture à trois niveaux est l'application du modèle plus général qu'est le multi-tiers.
- L'architecture logique du système est divisée en trois niveaux ou couches :
 - Couche présentation :
 - affichage, la restitution sur le poste de travail, dialogue avec l'utilisateur
 - couche métier ou traitement
 - mise en œuvre de l'ensemble des règles de gestion applicative
 - couche accès aux données
 - données destinées à être conservées sur la durée

14

Architecture 3-tier

■ Les trois couches



15

Architecture 3-tier

■ Couche Présentation (premier niveau)

- Correspond à la partie de l'application visible et interactive. On parle d'**Interface Homme Machine** (IHM). Peut être réalisée par une application graphique ou textuelle.
- Elle peut prendre de multiples facettes sans changer la finalité de l'application.
- Une même fonctionnalité métier pourra prendre **différentes formes de présentation** selon qu'elle se déroule sur Internet, sur un distributeur automatique de billets ou sur l'écran d'un chargé de clientèle en agence...
- Cette couche relaie les requêtes de l'utilisateur à destination de la couche métier, en retour elle lui **présente les informations** renvoyées par les traitements de cette couche. Il s'agit donc ici d'un assemblage de services métiers et applicatifs offerts par la couche inférieure.

16

Architecture 3-tier

- Couche Métier / Business (deuxième niveau)
 - partie fonctionnelle de l'application : implémente la « logique », et décrit les opérations que l'application **opère sur les données** en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation.
 - mises en œuvre des différentes **règles de gestion** et de **contrôle** du système
 - La couche métier offre des **services applicatifs et métier** à la couche présentation. Pour fournir ces services, elle s'appuie, le cas échéant, sur les données du système, accessibles au travers des services de la couche inférieure. En retour, elle **renvoie** à la couche présentation les **résultats** qu'elle a calculés.

17

Architecture 3-tier

- Couche Accès aux données (troisième niveau)
 - Données propres au système
 - Données destinées à **durer dans le temps**. Elles peuvent être stockées indifféremment dans de simples fichiers texte, ou eXtensible Markup Language (XML), ou encore dans un **base de données**.
 - Les services sont **mis à disposition de la couche métier**. Les données renvoyées sont issues du/des gisements de données du système.
 - Données gérées par un autre système
 - Les données peuvent aussi être gérées de manière externe. Elles ne sont pas stockées par le système considéré.

NB: La couche métier n'a pas à s'adapter à ces deux cas, ils sont transparents pour elle, et elle accède aux données de manière uniforme

18

Architecture 3-tier

■ Buts et Objectifs

- Allègement du poste de travail client
- Prise en compte de l'hétérogénéité des plates-formes (serveurs, clients, langages, etc.)
- Introduction de clients dits **légers**
- Amélioration de la sécurité des données, en supprimant le lien entre le client et les données. Le serveur a pour tâche, en plus des traitements purement métiers, de vérifier l'intégrité et la validité des données avant de les envoyer dans la couche de données
- Meilleure répartition de la charge entre différents serveurs d'application.

19

HTTP (HyperText Transfer Protocol)

- Il constitue le protocole de type question/réponse de transfert de fichiers hypertextes entre un serveur Web et un client Web.
- Exemple :

Requête

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Réponse

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
```

```
<html>
  <head>
    <title>Ma Page</title>
  </head>
  <body><p>Hello world!</p></body>
</html>
```

20

HTTP

- Format de la requête

<Méthode> <URL> HTTP/<Version>

[<Champ d'entête>: <Valeur>]

[<tab><Suite Valeur si >1024>]

ligne blanche

[corps de la requête]

- Exemple :

Requête

GET /index.html HTTP/1.1

Host: www.example.com

21

HTTP

- Format de la réponse

HTTP/<Version> <Status> <Commentaire Status>

[< Champ d'entête >: <Valeur>]

[<tab><Suite Valeur si >1024>]

Ligne blanche

Document

- Exemple :

Réponse

HTTP/1.1 200 OK

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

<html> <head> <title>Ma Page</title> </head>

<body><p>Hello world!</p></body>

</html>

22

HTTP

■ Principales Méthodes :

- **GET** : Pour demander une ressource est sans effet sur la ressource.
- **HEAD** : Ne demande que des informations sur la ressource, sans demander la ressource.
- **POST** : Envoyer de l'information, généralement entrée par l'utilisateur.
- **PUT** : Enregistrement du corps de la requête à l'URL indiqué
- **DELETE** : Efface la ressource spécifiée.

23

HTTP

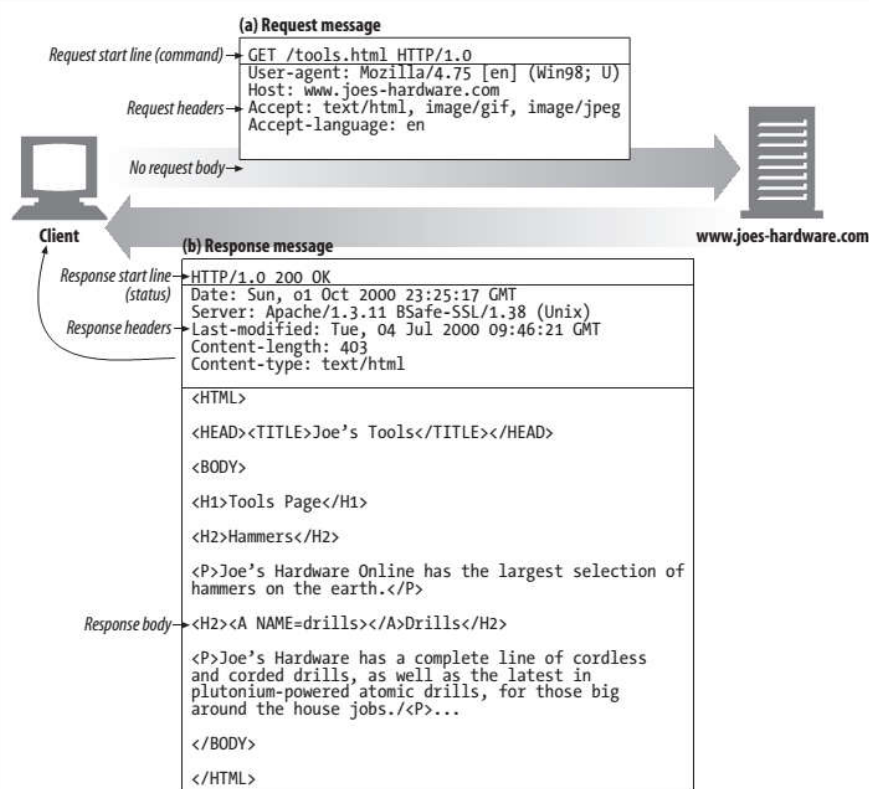
■ Statuts des réponses HTTP

- Le code (statut) est un entier sur trois chiffre dont le premier chiffre identifie la catégorie :
 - 1xx : Information
 - 2xx : Succès
 - 3xx : Redirection
 - 4xx : Erreur du client
 - 5xx : Erreur du serveur

24

HTTP

■ Exemple :



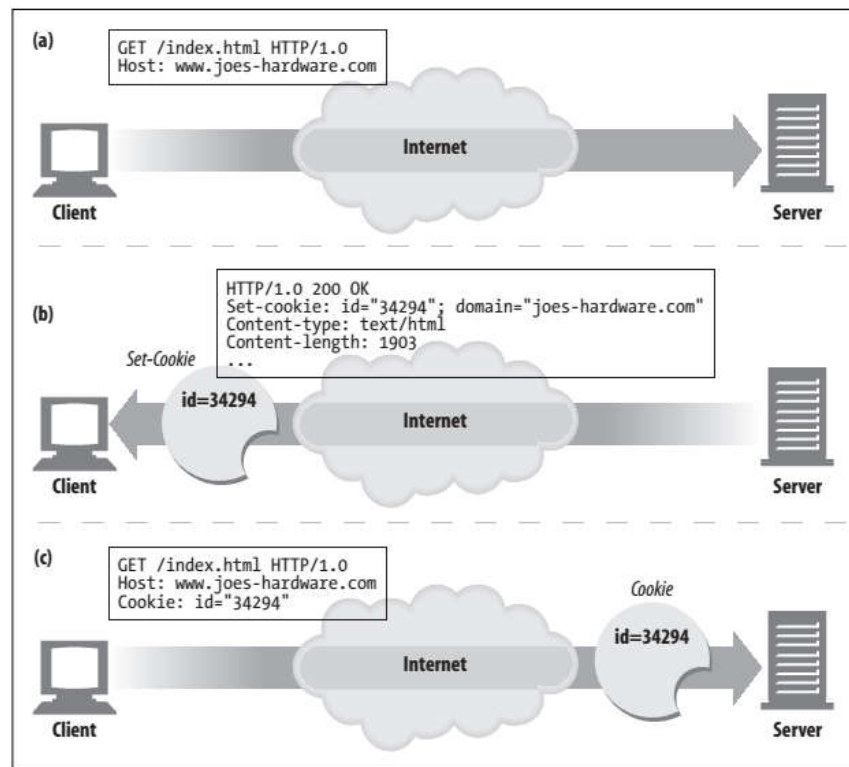
HTTP

■ Cookies

- information positionnée par le serveur sur le client
 - la durée de vie du cookie dépasse la session
- puis envoyé par le client à chaque requête

HTTP

■ Exemple :



27

URL (Uniform Resource Locator)

■ Définition

- L'URL est l'identificateur utilisé pour localiser une ressource sur Internet.

■ Syntaxe générale

- `<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`

28

URL (Uniform Resource Locator)

- Exemple :

- `http://www.exemple.com:80/chemin/vers/monfichier.html?clé1=valeur1&clé2=valeur2#QuelquePartDansLeDocument`


29

URL (Uniform Resource Locator)

- Exemple :

- `http://www.exemple.com:80/chemin/vers/monfichier.html?clé1=valeur1&clé2=valeur2#QuelquePartDansLeDocument`

`http://www.exe`



Le protocole


30

URL (Uniform Resource Locator)

- Exemple :

- `http://www.exemple.com:80/chemin/vers/monfichier.html?clé1=valeur1&clé2=valeur2#QuelquePartDansLeDocument`

`://www.exemple.com:80/`



Le nom de domaine


31

URL (Uniform Resource Locator)

- Exemple :

- `http://www.exemple.com:80/chemin/vers/monfichier.html?clé1=valeur1&clé2=valeur2#QuelquePartDansLeDocument`

`ple.com:80/chemin/`



Le port

32

URL (Uniform Resource Locator)

- Exemple :

- `http://www.exemple.com:80/chemin/vers/monfichier.html?clé1=valeur1&clé2=valeur2#QuelquePartDansLeDocument`

`30/chemin/vers/monfichier.html?key`

Le chemin vers le fichier

33

URL (Uniform Resource Locator)

- Exemple :

- `http://www.exemple.com:80/chemin/vers/monfichier.html?clé1=valeur1&clé2=valeur2#QuelquePartDansLeDocument`

`ml?clé1=valeur1&clé2=valeur2#Ql`

Les paramètres

34

URL (Uniform Resource Locator)

- Exemple :

- `http://www.exemple.com:80/chemin/vers/monfichier.html?clé1=valeur1&clé2=valeur2#QuelquePartDansLeDocument`

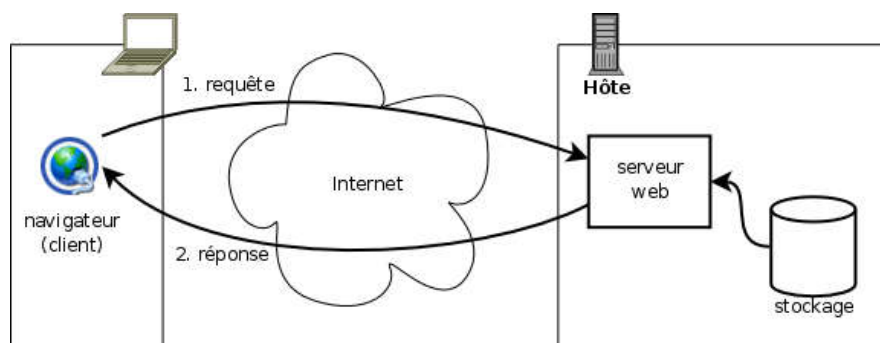
`ur2#QuelquePartDansLeDocument`

Une ancre

35

Page Web statique

- C'est une page web constituée d'un fichier texte contenant du code HTML et éventuellement des images et des liens vers d'autres documents.



36

Technologies

- Le développement de plusieurs technologies a permis au Web d'agir de manière plus intelligente.
 - Les technologies côté client : qui complètent HTML et permettent aux clients web de faire des choses plus sophistiquées que l'affichage de documents statiques
 - Les technologies côté serveur : qui permettent au serveur de construire des pages web à la volée et fonctionnent généralement avec une base de données.

37

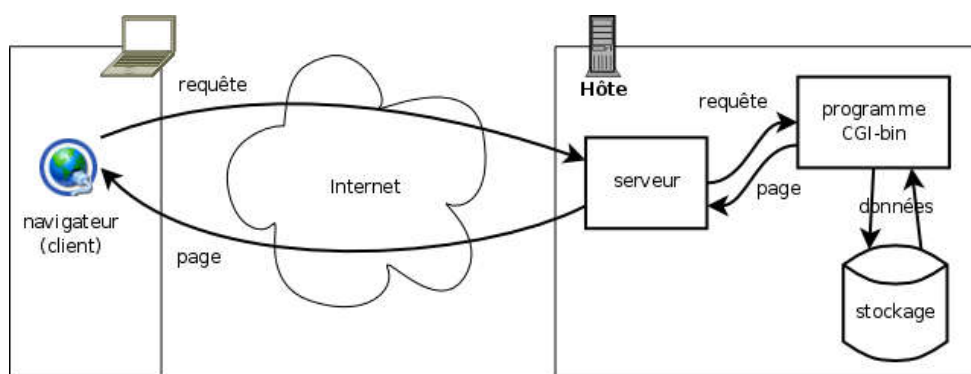
Technologies côté client

- Plusieurs technologies possibles :
 - **JavaScript** : langage de script interprété par le navigateur et permettant de manipuler l'arborescence du document(DOM) ;
 - **Applet Java** : mini application Java permettant d'utiliser toute la puissance du langage Java (API, structures de données, ...)
 - **ActionScript** : langage de script compatible avec JavaScript et permettant de réaliser des animations Flash ...

38

Technologies côté serveur

- Par le biais du CGI (Common Gateway Interface)
 - Les programmes ainsi utilisés au travers de l'interface CGI sont appelés *programmes CGI* ou *CGI-bin*.



39

Technologies côté serveur

- Par le biais du CGI (Common Gateway Interface)
 - Exemple :

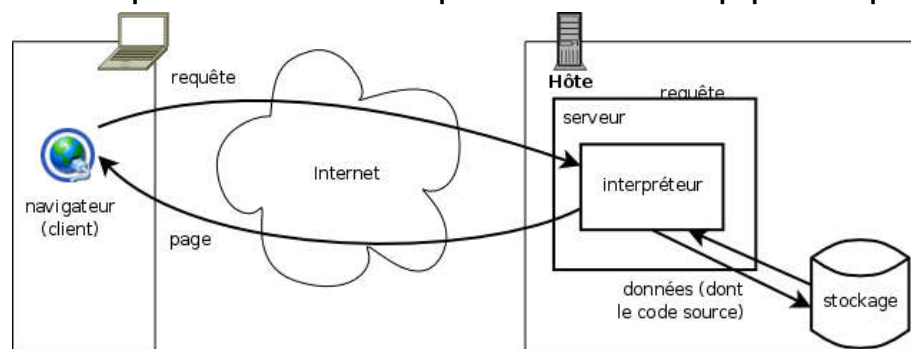
```

# include ...
void main ()
{
/* ... */
printf (" Content-type : text / html \n " );
printf ( " Connexion : close \n \n " );
/* ... */
printf (" <h1 > Hello World! </ h1 >\n " );
/* ... */
}
  
```

40

Technologies côté serveur

- Par le biais des modules
 - certains serveurs ont intégré des modules interprétant des langages de programmation tels que Perl, Python, PHP.
 - => l'interprétation des scripts est beaucoup plus rapide.



41

Page Web dynamique plus interactive

- AJAX (Asynchronous JAVascript and Xml)
 - Ajax apporte une technique qui permet d'appeler le serveur en arrière-plan via JavaScript et d'obtenir des données complémentaires à la demande afin d'actualiser la page sans déclencher son rechargement intégral

42