Architecture des réseaux et des systèmes informatiques répartis

Théo Pierron

Polytech 4A Apprentissage

HTTP

World Wide Web

WWW = Architecture permettant l'échange de documents (pages web,...) liés entre eux entre des machines connectées sur le réseau Internet.

World Wide Web

WWW = Architecture permettant l'échange de documents (pages web,...) liés entre eux entre des machines connectées sur le réseau Internet. 3 ingrédients :

- Un langage : HyperTextMarkingLanguage « quoi »
- Un protocole : HyperTextTransferProtocol « comment »
- Un lieu : UniformResourceLocator « où »

URL

```
protocole://login:mdp@serveur:port/chemin/fichier?argument#balise
```

- protocole : http, ftp,... « comment ».
- login/mdp : facultatif, transmis en clair
- serveur : le nom de domaine
- port : 80 par défaut
- /chemin/fichier : comme sur sa machine
- argument : à transmettre au serveur
- balise : où positionner le navigateur

URL

```
protocole://login:mdp@serveur:port/chemin/fichier?argument#balise
```

- protocole : http, ftp,... « comment ».
- login/mdp : facultatif, transmis en clair
- serveur : le nom de domaine
- port : 80 par défaut
- /chemin/fichier : comme sur sa machine
- argument : à transmettre au serveur
- balise : où positionner le navigateur

Absolue ou relative.

HTTP

- Client/serveur
- Basé sur TCP
- Traitement parallèle des clients

Côté client

Un navigateur Web :

- Cherche la page demandée

Côté client

Un navigateur Web :

- Cherche la page demandée
  - Analyse l'URL $\rightarrow$ nom du serveur
  - DNS $\rightarrow$ IP du serveur
  - Établit une connexion TCP
  - Fabrique une requête et l'envoie
  - Récupère la réponse

Côté client

Un navigateur Web :

- Cherche la page demandée
  - Analyse l'URL $\rightarrow$ nom du serveur
  - DNS $\rightarrow$ IP du serveur
  - Établit une connexion TCP
  - Fabrique une requête et l'envoie
  - Récupère la réponse
- Interprète le langage HTML pour l'affichage
- Va chercher les images ou autres ressources (animations, javascript...)
- Affiche la page finale

Côté serveur

Un serveur Web :

- Écoute les requêtes
- Vérifie la validité (syntaxe des requêtes, disponibilité des documents,...)
- Envoie du texte, du code, un message d'erreur, une demande d'authentification

Côté serveur

Un serveur Web :

- Écoute les requêtes
- Vérifie la validité (syntaxe des requêtes, disponibilité des documents,...)
- Envoie du texte, du code, un message d'erreur, une demande d'authentification

1 requête = 1 connexion

Améliorations

- Utilisation de formulaires pour envoyer des informations au serveur.
- Pages dynamiques
- Sessions permanentes
- Pas que des fichiers texte
- Utilisation d'un cache

Fonctionnement

Le client envoie

```
GET <url> HTTP/<version>
```

puis une ligne vide.

Le client envoie

```
GET <url> HTTP/<version>
```

puis une ligne vide.

Le serveur répond

```
HTTP/<version> <code> <message>
```

puis une ligne vide
puis le contenu de la page (si la requête a réussi).

Types de réponse

- 1xx : requête reçue, traitement en cours
- 2xx : succès
- 3xx : redirection
- 4xx : erreur du client
- 5xx : erreur du serveur

Exemple : 200 OK, 404 NOT FOUND, 418 I'M A TEAPOT

Plus d'informations : l'en-tête

Le client et le serveur peuvent envoyer plus d'informations, par exemple :

- leur encodage
- la date dedernière modification
- le type des fichiers
- la taille de la réponse
- ...

$\rightarrow$ requête HEAD : demande seulement l'en-tête de la réponse.

Plus d'informations : les formulaires

But : envoyer au serveur les informations que le client a fournies sur la page.

Plus d'informations : les formulaires

But : envoyer au serveur les informations que le client a fournies sur la page.
Idée : utiliser l'URL

```
GET /dossier/page.html?champ1=info1& champ2=info2
```

Plus d'informations : les formulaires

But : envoyer au serveur les informations que le client a fournies sur la page.
Idée : utiliser l'URL

```
GET /dossier/page.html?champ1=info1& champ2=info2
```

Problèmes :

- Passage des arguments en clair
- Taille des URL limitée
- Sauvegarder l'URL = sauvegarder les informations aussi

$\rightarrow$ requête POST : les informations sont transmises dans le corps de la requête.

D'autres requêtes

- PUT : écrire une page (inverse du GET)
- DELETE : supprime la page
- TRACE : sert au débug
- OPTIONS : fournit des informations sur les requêtes supportées par le serveur

D'autres requêtes

- PUT : écrire une page (inverse du GET)
- DELETE : supprime la page
- TRACE : sert au débug
- OPTIONS : fournit des informations sur les requêtes supportées par le serveur

Tout est optionnel sauf GET et HEAD.

Utilisation des en-têtes

- À l'origine, HTTP sert à échanger des fichiers texte
- Maintenant, les pages contiennent des objets complexes !

Durée des connexions

`Connection:close/keep-alive`

Type des fichiers échangés

- Format MIME (Multi-purpose Internet Mail Extensions)
- Un type : `text`, `image`, `audio`, ...
- Un sous-type : `text/html`, `image/gif`, `application/pdf`, ...

Type des fichiers échangés

- Format Multi-purpose Internet Mail Extensions
- Un type : `text`, `image`, `audio`, ...
- Un sous-type : `text/html`, `image/gif`, `application/pdf`, ...

Le serveur envoie `Content-type:<type>` dans son en-tête.

Type des fichiers échangés

- Format MIME (Multi-purpose Internet Mail Extensions)
- Un type : `text`, `image`, `audio`, ...
- Un sous-type : `text/html`, `image/gif`, `application/pdf`, ...

Le serveur envoie `Content-type:<type>` dans son en-tête.
Le client sait comment lire les données reçues.

Serveur cache

Objectif : ne pas renvoyer une page si elle n'a pas été modifiée.

Serveur cache

Objectif : ne pas renvoyer une page si elle n'a pas été modifiée.

- Le client garde une copie de chaque page reçue.
- Utilise

`If-modified-since:<date>`

- Si le serveur renvoie une page vide, on affiche la page en cache.

Serveur cache

Objectif : ne pas renvoyer une page si elle n'a pas été modifiée.

- Le client garde une copie de chaque page reçue.
- Utilise

`If-modified-since:<date>`

- Si le serveur renvoie une page vide, on affiche la page en cache.

Cache géré par le navigateur ou par le réseau (proxy).

Identifier le client

Serveur Web = sans mémoire (pas de session)

Identifier le client

Serveur Web = sans mémoire (pas de session)
Comment garder la trace du client ?

- IP ? $\rightarrow$ non (ordis partagés, utilisation nomade)
- Cookies !

Cookies

Cookie =

- petit fichier (4Ko max) écrit par le serveur chez le client.
- ne contient que de l'information (pas de code).

Exemple : identifiant d'un client, code d'accès.

Cookies

Cookie =

- petit fichier (4Ko max) écrit par le serveur chez le client.
- ne contient que de l'information (pas de code).

Exemple : identifiant d'un client, code d'accès.
Utilisation détournée :

- Collecte d'information
- Piratage

Fonctionnement

Première réponse HTTP :

`Set-cookie:nom=valeur;expires=;path=;domain=;secure`

- nom/valeur = contenu du cookie
- expires = date de validité
- path = cookie renvoyé pour chaque requête d'une page dans le dossier indiqué

Fonctionnement

Première réponse HTTP :

```
Set-cookie:nom=valeur;expires=;path=;domain=;secure
```

- nom/valeur = contenu du cookie
- expires = date de validité
- path = cookie renvoyé pour chaque requête d'une page dans le dossier indiqué

À chaque requête, le client vérifie ses cookies et ajoute le contenu de ceux qui correspondent dans l'en-tête.

Pages dynamiques

But : Générer automatiquement les pages.
Techniques :

- CommonGatewayInterface
- Script serveur
- Script client

CGI

CGI =

- Fichier sur le serveur
- Quand un utilisateur le GET, il s'exécute sur le serveur et génère une page web qui est renvoyée au client.
- Utilise les entrées du client
- Peut être complexe (appel à une BDD)

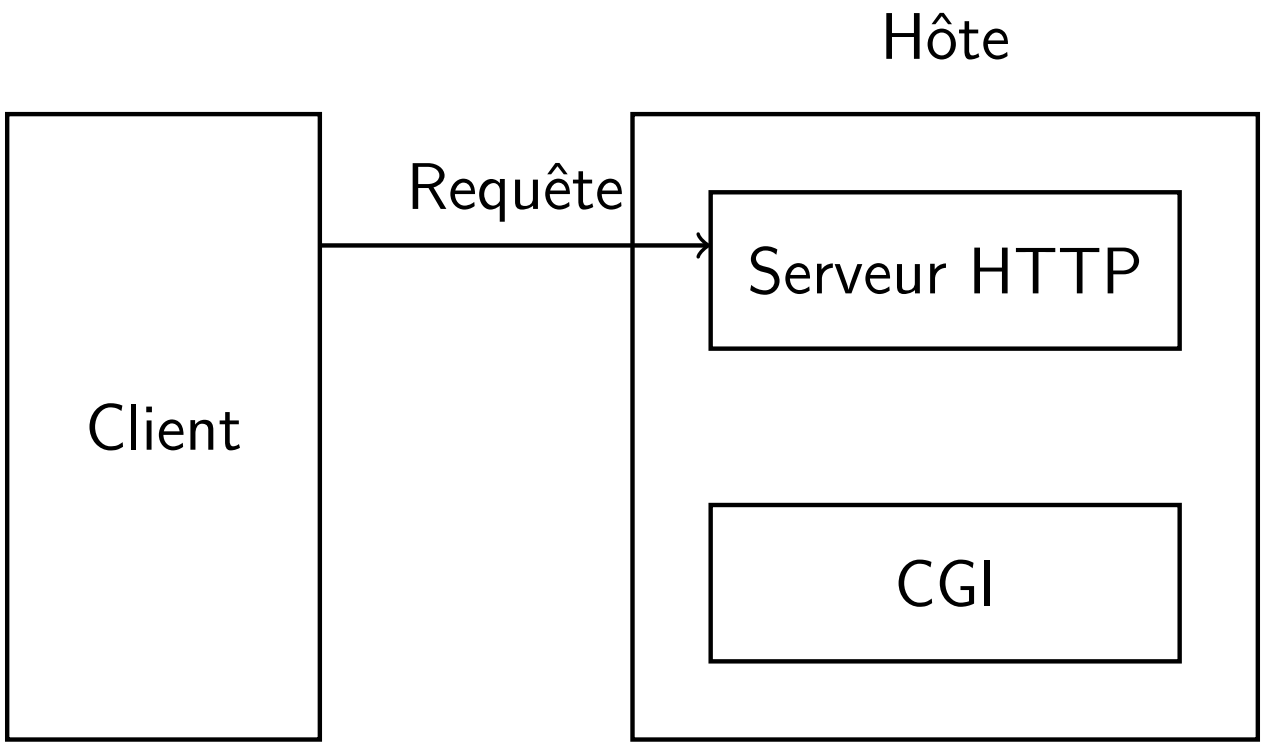
CGI

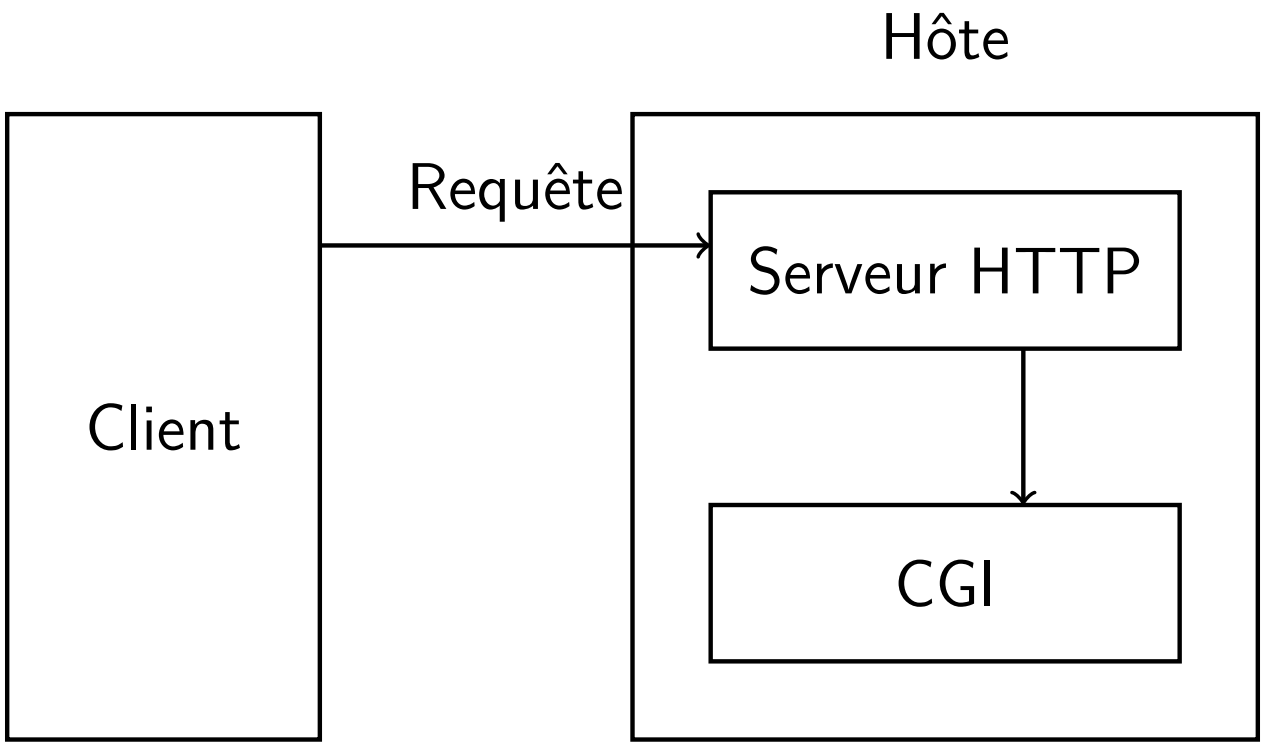
CGI =

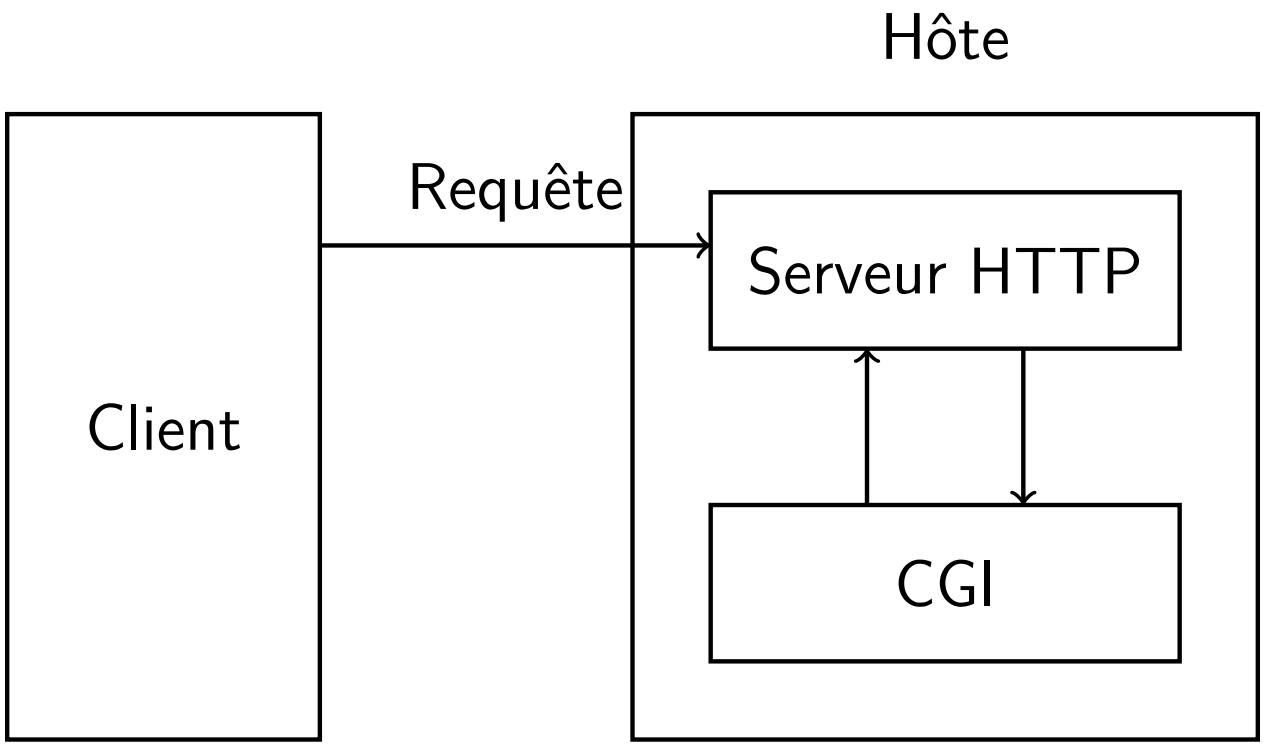
- Fichier sur le serveur
- Quand un utilisateur le GET, il s'exécute sur le serveur et génère une page web qui est renvoyée au client.
- Utilise les entrées du client
- Peut être complexe (appel à une BDD)

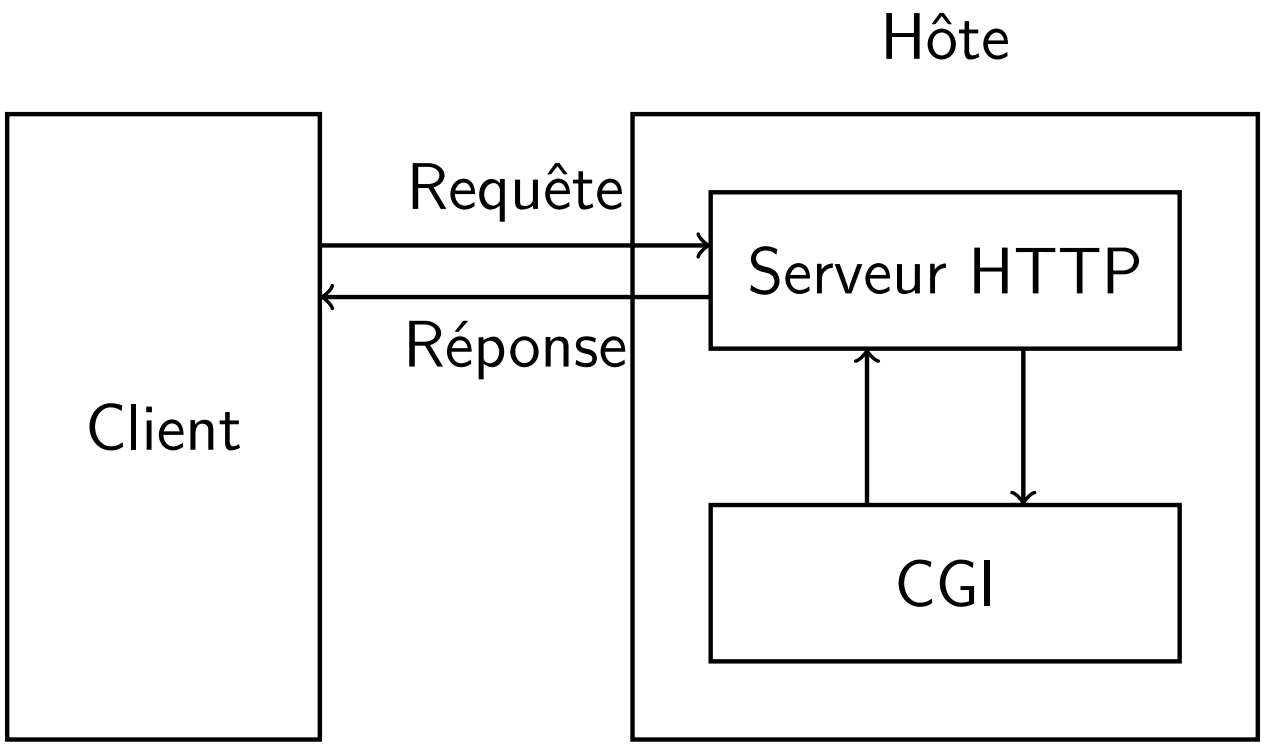
Attention !

- Permet de faire exécuter n'importe quoi par le serveur http
- Risque de surcharge du serveur
- Le client attend pendant l'exécution


Hôte
Requête
Client
Serveur HTTP
CGI


Hôte
Requête
Client
Serveur HTTP
CGI

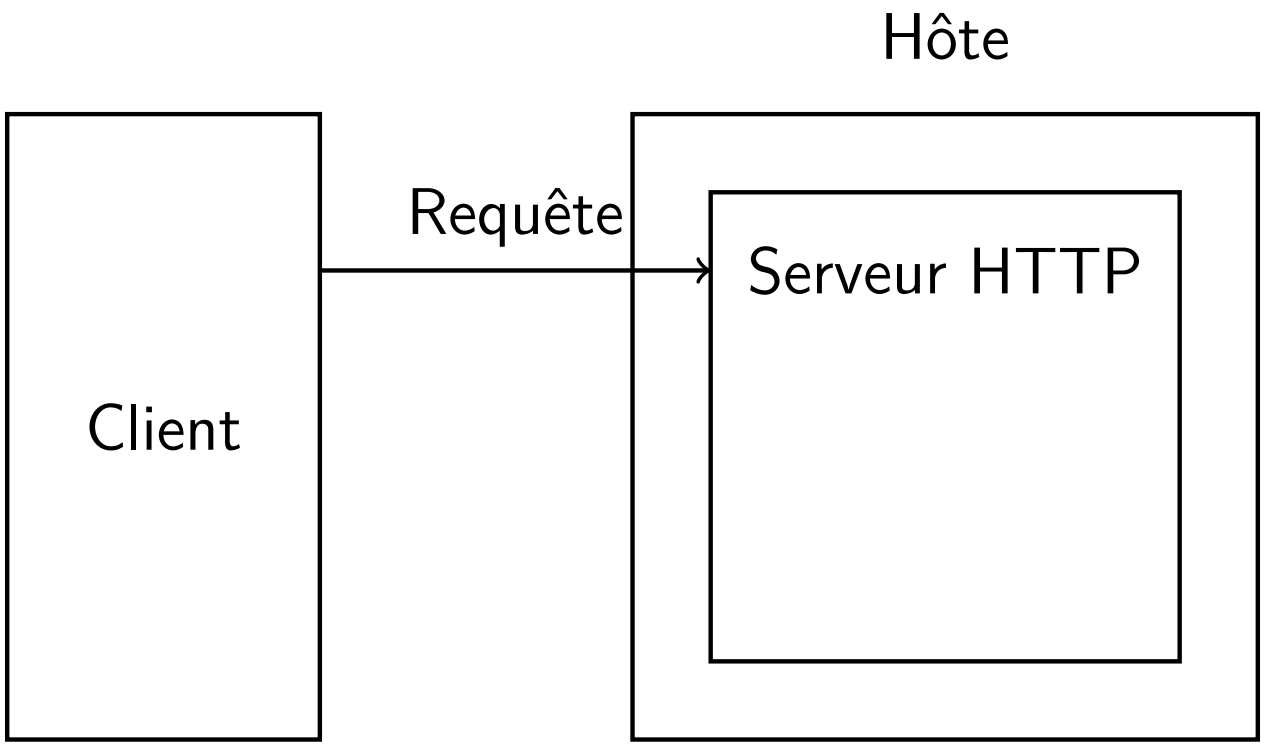

Hôte
Requête
Client
Serveur HTTP
CGI

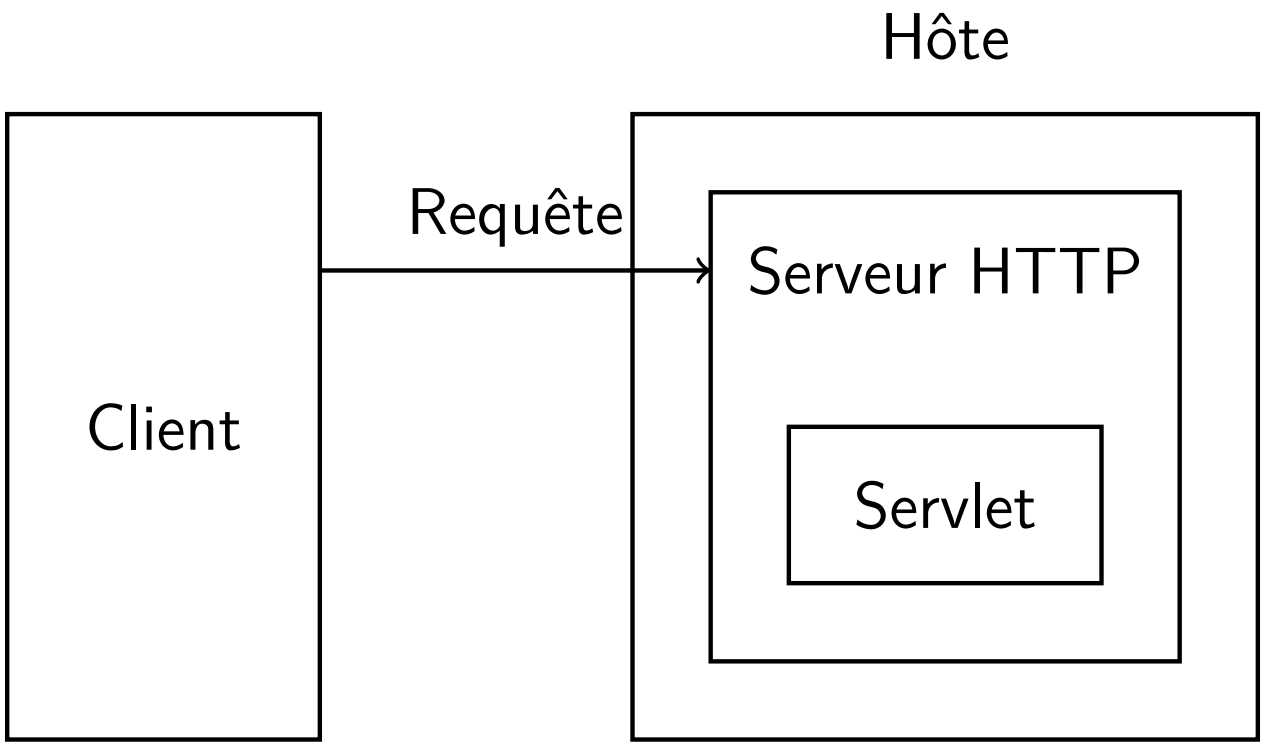

Hôte
Client
Requête
Réponse
Serveur HTTP
CGI

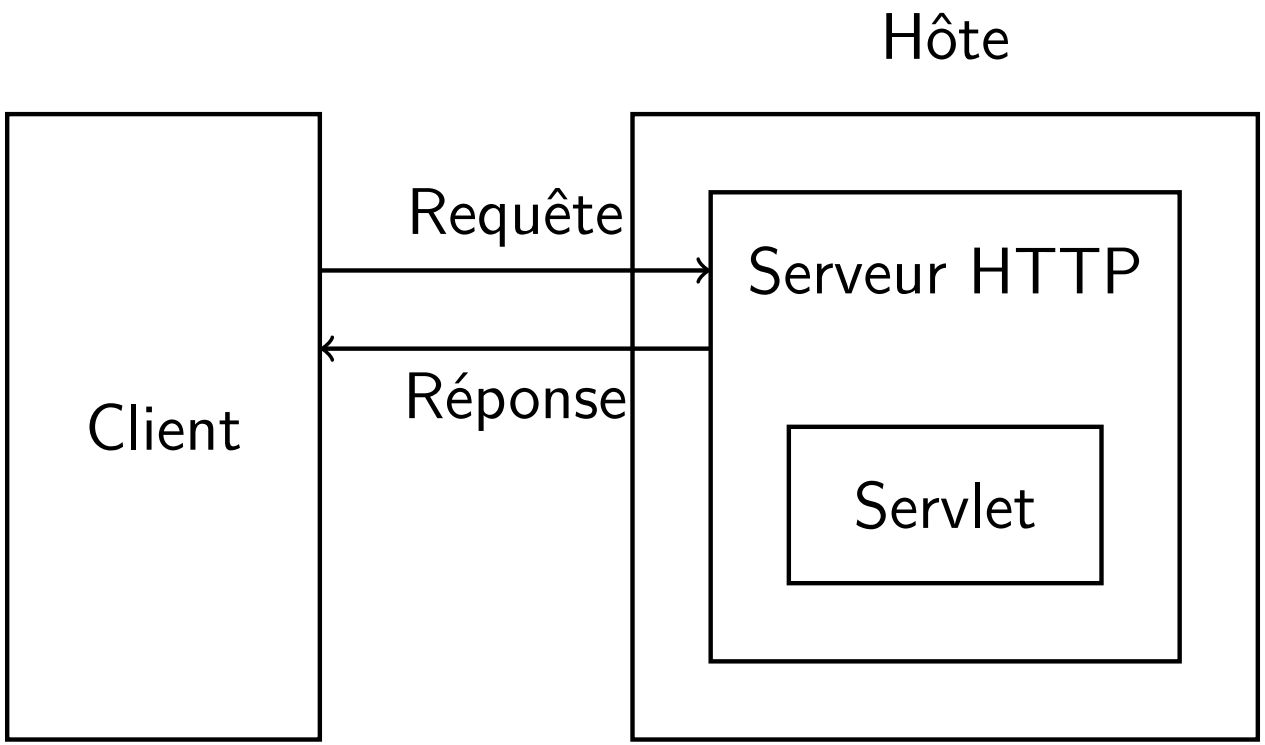
Script serveur/Servlets

Le serveur :

- Détecte une balise spéciale dans une page (dépend du serveur)
- Exécute le code qu'elle contient
- La remplace par le résultat
- Transmet la page obtenue


Hôte
Requête
Serveur HTTP
Client

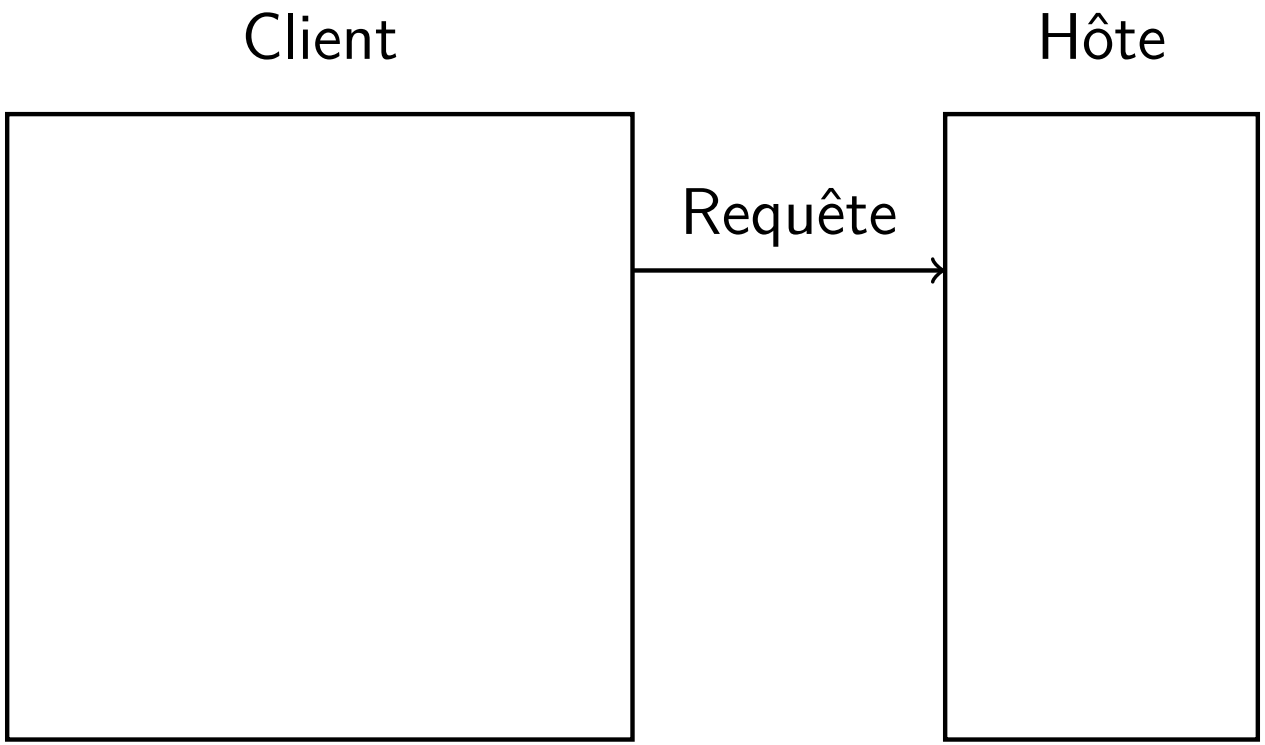

Hôte
Requête
Client
Serveur HTTP
Servlet

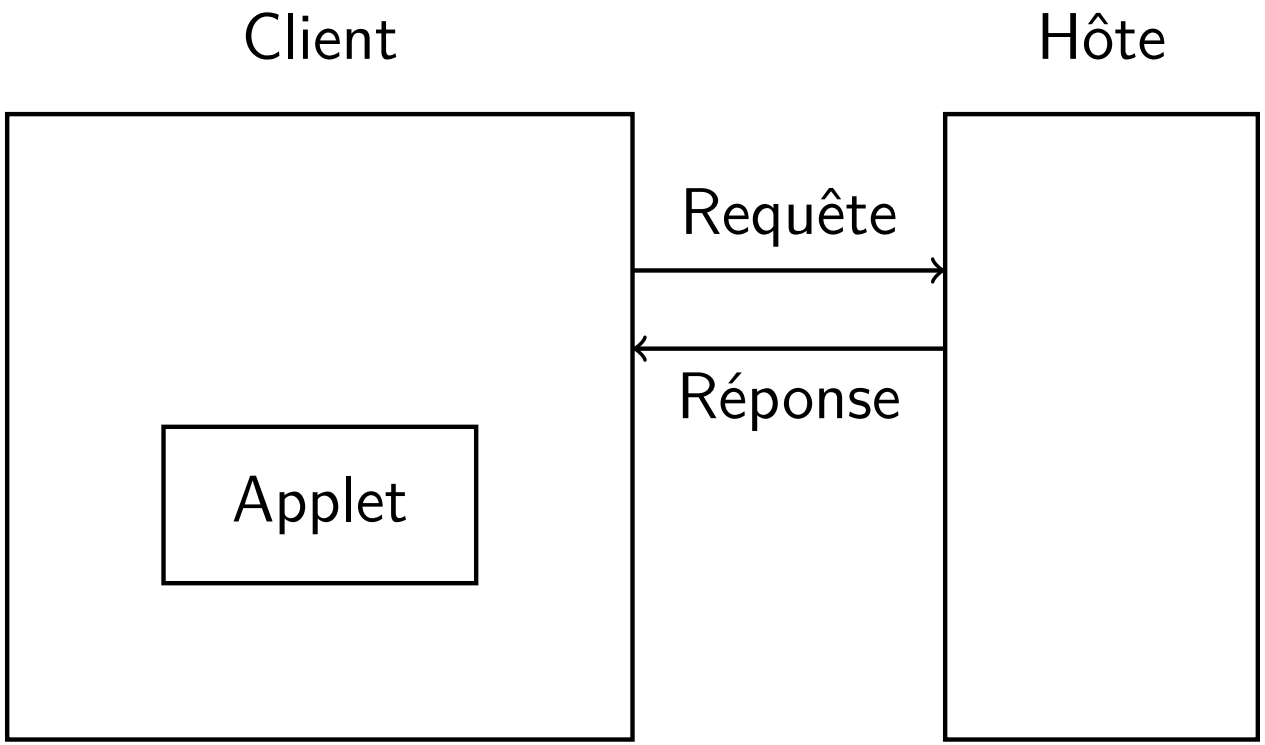

Hôte
Client
Requête
Réponse
Serveur HTTP
Servlet

Script client/Applets

Applet =

- Application indépendante
- Généralement du code java précompilé
- Envoyé au client, puis exécuté sur la machine du client


Client
Hôte
Requête


Client
Hôte
Requête
Réponse
Applet