
JAKARTA EE
DÉVELOPPEMENT WEB
Par Joachim Zadi – Mai 2019
SAGEFOX

Les pages Web dynamiques

Présentation

Cette étude porte uniquement sur les principes généraux des pages Web dynamiques. Une fois que nous aurons bien maîtriser ces différents concepts, nous pourrons aborder les cas pratiques en se servant de la plateforme Java. Java met en œuvre un certain nombre de technologies pour la mise en œuvre de ces pages Web dynamiques, comme les servlets, les JSP, les EJB (Entreprise Java Bean). Ces différentes technologies seront présentées lors des études suivantes. Toutefois, avant de les traiter, je pense qu'il convient de connaître quelques éléments qui permettent de comprendre ce que sont les services et les applications Web.

CHAPITRES TRAITÉS :

1. *Les services sur Internet et le modèle client-serveur*
2. *Protocole de communication*
3. *Protocole HTTP*
4. *Le type MIME*
5. *Programme CGI*

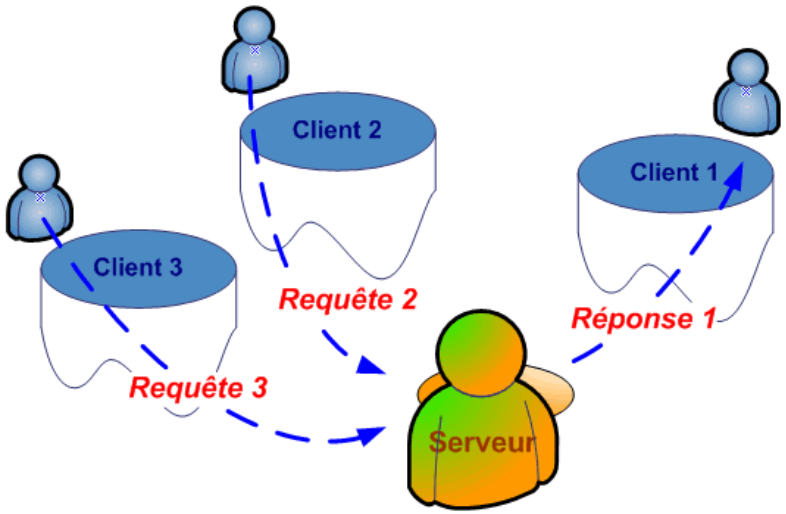
Les services sur Internet et le modèle client-serveur

Les sites Web dynamiques existent parce qu'Internet existe. La structure et le fonctionnement d'un site dynamique sont donc fortement liés au fonctionnement d'Internet et au mode de communication des ordinateurs. Les applications développées pour les sites Internet repose essentiellement sur le modèle client-serveur.

Présentation du client et du serveur

Le modèle client-serveur a été conçu bien avant Internet. Le concept a été mis en place lorsque les ordinateurs ont pus être connectés pour former un réseau local et ainsi établir un dialogue entre eux. Pour bien comprendre le principe, nous allons le décrire de façon imagée.

Les termes client et serveur ne sont pas anodins. Ils sont issus du monde réel. En effet, le fonctionnement client-serveur se rapproche tout naturellement des rapports existants entre le client et le serveur d'un restaurant.



Lorsque vous allez au restaurant, vous êtes le client et vous souhaitez commander un menu. Pour cela, vous appelez le serveur « requête ». Ce dernier doit gérer plusieurs tables « clients ». Il répond au fur et à mesure, à la demande chaque client en fonction de ce qui est disponible. Il établit donc une relation entre les clients de la salle et les ressources disponibles en cuisine.

Transposé dans le monde informatique, le concept client-serveur est très proche de cette description. Le client est une application qui s'exécute qui s'exécute sur un ordinateur personnel. Le serveur est une autre application qui gère des ressources partagées, et qui est programmé pour rendre un service donné en réponse à une requête qui lui est adressé.

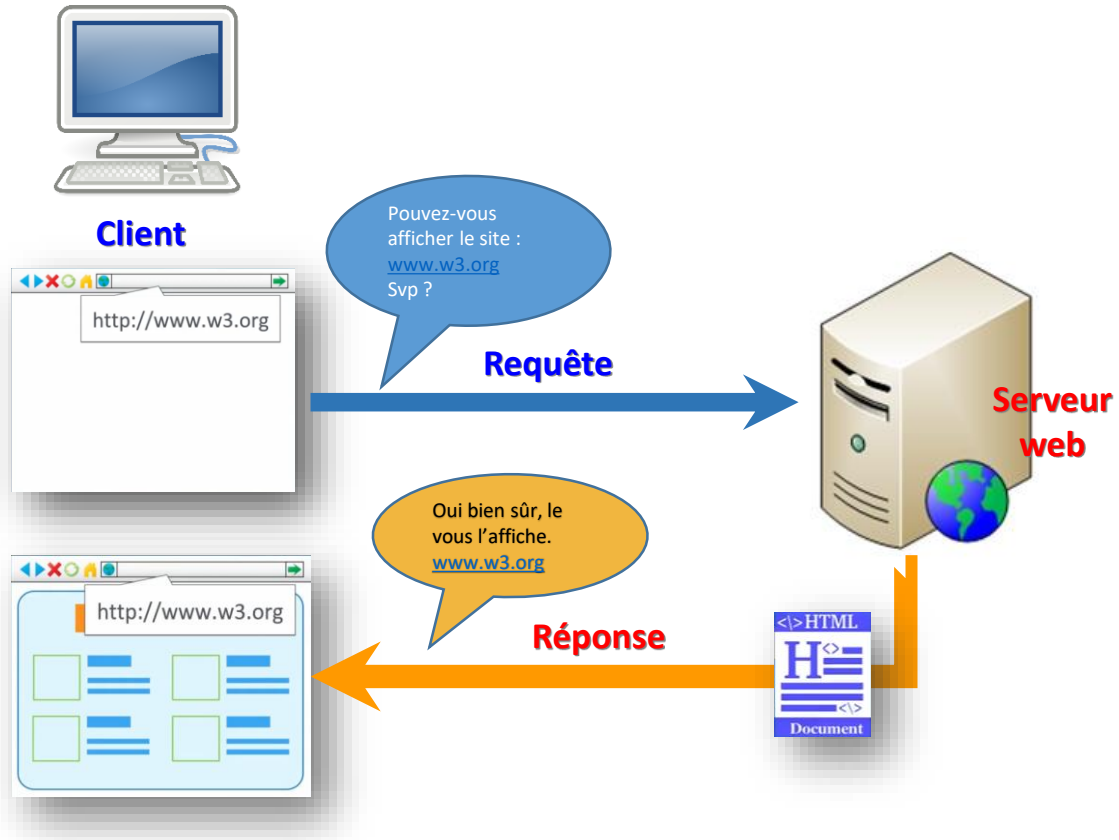
Les pages Web dynamiques

Service Web

Le Web est un exemple tout choisi d'architecture client-serveur : les sites sont animés par des serveurs qui rendent toujours le même type de services aux clients que sont les navigateurs. Quel service au juste ? Le serveur Web attend qu'on lui demande des données « statiques ou résultant d'un traitement ». En réponse à une requête, il envoie le contenu des données requises. Le navigateur de son côté est client du serveur auquel il a envoyé une requête. Le navigateur présente à l'utilisateur une réponse une fois les données téléchargées.

- *style "http://www.unsite.net" (URL).*
- *Le serveur Web contacté répond au client en affichant l'ensemble des informations stockées et organisées sur son disque dur à l'URL donné.*

Un client est une application qui se connecte à un autre ordinateur pour obtenir ou modifier des informations à l'aide de requêtes. Un serveur est une application située sur un ordinateur « très puissant » capable de gérer un très grand nombre de requêtes simultanément. Un serveur est toujours en attente de requêtes.



D'autres Services

Il existe différent types de serveurs : serveur de messagerie, serveur d'accès distant, serveurs de transfert de fichiers, serveurs de base de données. Parallèlement, il existe différentes applications clientes pour consulter ou modifier les ressources des serveurs. Pour que tous les clients et tous les serveurs arrivent à travailler ensemble, des règles de communication ont été définies.

Protocole de communication

Dans le modèle client-serveur, tout est construit autour de la communication entre le client et le serveur. Sans cette communication, il ne peut y avoir de requête ni de réponse. La communication s'établit en suivant un certain nombre de règles que l'on appelle protocoles de communications.

Ces protocoles sont en réalité des modèles qui décrivent l'organisation et la transmission des données numériques lors d'un échange entre le client et le serveur. Les règles qui en découlent sont respectées à la fois par les applications clientes et par les applications serveur. Elles ont fait l'objet d'une norme, de façon à ce que toute application orientée sur un service « messagerie électronique, Web, etc. » soit capable de comprendre un message provenant d'une autre application orientée sur le même service.

Choisir un protocole pour communiquer

Nous venons de le voir, la clé de voute qui permet à une architecture client-serveur de fonctionner, sans d'ailleurs que ni le client ni le serveur ne sachent comment ils fonctionnent l'un l'autre, est l'utilisation d'un protocole commun. Un protocole établit la norme que doit respecter un client pour communiquer avec un serveur ;

1. *Comment établir une communication entre un client et un serveur ;*
2. *Ce qu'un client peut envoyer comme requête à un serveur ;*
3. *Ce qu'un serveur répondra à la requête du client.*

Le protocole le plus utilisé pour communiquer avec un serveur Web sur Internet est le protocole HTTP (Hyper Text Transfer Protocole). Il définit les règles de communication entre un client (navigateur) et un serveur Web.

Les pages Web dynamiques

Adresse IP et port, point de rendez-vous des Serveurs Internet

Un serveur Internet est accessible par un nom de machine hôte « par exemple oracle.com, www.google.fr ». Chaque nom correspond à une adresse IP, qui est l'identifiant représentant la machine hôte où tourne un serveur « moi même, je possède un nom, et pour communiquer avec moi et recevoir des lettres, le facteur a besoin de connaître mon adresse ». Ce nombre de 32 bits est généralement noté sous la forme d'une suite de 4 nombres de 8 bits chacun séparés par des points « par exemple 127.0.0.1 ». Chaque machine relié à Internet « client et serveur » est identifié une adresse IP qui la représente sur le réseau.

Pour distinguer les applications serveurs (les services) qui tournent simultanément sur une même machine hôte, un deuxième niveau d'identification est nécessaire, qui est donné par le numéro de port, codé sur 16 bits et donc compris entre 0 et 65535. Le fait de faire référence à un port indique quel est le service que nous désirons utiliser.

Pour éviter que ce soit l'anarchie, des numéros de port ont été normalisé par rapport au services que nous utilisons le plus couramment. Ainsi, le service Web « HTTP » utilise le numéro de port 80. Le service FTP utilise le numéro de port 21, etc. Ces numéros de port étant normalisés, ils constituent donc les valeurs par défaut, et du coup, lorsque nous utilisons le service associé, il n'est pas nécessaire de préciser le numéro. Implicitement, c'est le bon numéro de port qui est pris.

Exemple d'une requête HTTP vers une URL

La requête la plus simple du protocole HTTP est formé de GET suivi d'une URL qui pointe sur des données « fichier statiques, traitement dynamique... ». Elle est envoyée par un navigateur quand nous saisissons directement une URL dans le champ d'adresse du navigateur. Le serveur HTTP répond en renvoyant les données demandées.

Les pages Web dynamiques

1. *En tapant l'URL d'un site, l'internaute envoie (via le navigateur) une requête au serveur.*
2. *Une connexion s'établit entre le client et le serveur sur le port 80 (port par défaut d'un serveur Web).*
3. *Le navigateur envoie une requête demandant l'affichage d'un document. La requête contient entre autres la méthode (GET, POST, etc.) qui précise comment l'information est envoyée.*
4. *Le serveur répond à la requête en envoyant une réponse HTTP composée de plusieurs parties, dont :*
   - *l'état de la réponse, à savoir une ligne de texte qui décrit le résultat du serveur (code 200 pour un accord, 400 pour une erreur due au client, 500 pour un erreur due au serveur) ;*
   - *les données à afficher.*
5. *Une fois la réponse reçue par le client, la connexion est fermée. Pour afficher une nouvelle page du site, une nouvelle connexion doit être établie.*

Les méthodes POST et GET

Les méthodes POST ou GET déterminent la façon dont est envoyée la requête au serveur. En effet, il existe plusieurs façons de transmettre une requête, notamment lorsque celle-ci contient des valeurs « paramètres »qui permettront au serveur de faire des réponses différenciées « pages Web dynamiques ».
Classiquement, la transmission des valeurs via le navigateur s'effectue par la mise en place d'une chaîne de données à la suite de l'URL. Ce type de transmission est utilisée par la méthode GET. Par exemple, L'URL :

http://www.unsite.net/rechercher?nom=Lagafe&prénom=Gaston

indique au serveur qu'il doit afficher la page associée aux paramètres nom et prénom qui ont pour valeur Lagafe et Gaston respectivement.

Les pages Web dynamiques

La syntaxe générale de l'URL correspondant à la méthode GET est la suivante :

1. *Localisation du site (donc du serveur) : http://www.unsite.net/*
2. *Programme à lancer côté serveur pour traiter la requête désirée : rechercher*
3. *Un point d'interrogation ? : opérateur pour séparer le programme des paramètres qui vont servir au traitement.*
4. *Paramètres qui servent au traitement, chacun est séparé du suivant par l'opérateur &.*
5. *Chaque paramètre possède un nom (prénom) suivi de l'opérateur = suivi ensuite de la valeur (Gaston) que prend ce paramètre.*

Avec la méthode GET, les informations sont donc stockées dans l'URL. Ce mode de transmission est le plus simple de mise en oeuvre. Par contre, il présente l'inconvénient de rendre visibles les données sensibles telles qu'un mot de passe ou un code de carte banquaire. En outre, la longueur de la chaîne transférée est limitée.

Si le nombre de paramètres est important, ou si les valeurs sont confidentielles, il est conseillé d'utiliser la méthode POST. En effet, la méthode POST résout ces deux problèmes en envoyant les valeurs des paramètres dans le corps même de la requête et non via l'URL. De cette façon, aucune valeur n'apparaît dans l'URL. Cette dernière reste fixe quelles que soient les options choisies par l'internaute.

Par contre, la récupération des valeurs paraît un peu plus délicate puiqu'il faut scruter la requête elle-même. Ceci dit, nous verrons que dans Java, cela ne pose aucun problème puisque des objets sont spécialisés dans ce domaine. Il suffira d'appeler la méthode adéquate de l'objet concerné.

La notion de session

Ainsi, lorsque le serveur a traité une requête et envoyé sa réponse au client, la connexion entre le serveur et le client est clôturée. Le serveur perd la trace du client. Si ce dernier émet une nouvelle requête, le serveur la traite, ne sachant pas qu'il a déjà communiqué avec ce client.

La communication entre un client et un serveur n'est pas continue. On appelle ce mode de communication, le mode non connecté.

Cette façon de communiquer a des conséquences importantes sur le développement de sites commerciaux où le serveur doit enregistrer les achats du client. En effet, lorsque l'internaute achète sur Internet, il sélectionne chaque objet qu'il souhaite acheter l'un après l'autre. A chaque objet sélectionné correspond une requête distincte de la précédente « avec systématiquement une ouverture et une fermeture de la connexion ».

Le serveur, ou plus exactement le programme exécuté sur le serveur, doit donc se souvenir du client afin de regrouper toutes les requêtes d'achat et de les identifier comme appartenant au même internaute.

Pour réaliser cette performance, il convient de mettre en place un suivi de session qui permet d'enregistrer toutes les requêtes d'un même client sous un numéro d'identification.

L'URL, notation unique d'une ressource sur Internet

Une URL « Uniform Resource Locator » est une chaîne de caractères qui désigne une ressource sur un serveur. Elle est de la forme :

protocole://hote:port/chemin/vers/ressource

où

- *protocole : représente le protocole d'accès au serveur désigné. Par exemple, http, ftp ou file ;*
- *hôte : est le nom ou l'adresse IP de la machine hôte où le serveur tourne ;*
- *port : est le numéro du service avec lequel nous communiquons ;*
- *chemin/vers/ressource : est le chemin d'accès à la ressource sur le serveur.*

Une ressource représente une information ou un programme mis à disposition. Ce peut être un fichier, une image, une application, à vrai dire tout ce qui pourrait être utilisé d'une manière ou d'une autre.

Les numéros des ports des protocoles FTP et HTTP sont respectivement et implicitement les numéros 21 et 80 dans une URL s'ils ne sont pas mentionnés. Voici quelques exemples :

http://www.google.fr:80/ équivalent à http://www.google.fr/
http://magasin-en-ligne:80/jsp.informatique/Vitrine/index.jsp?page=cahier
http://www.autre.com/index.html

Protocole HTTP

Dans ce chapitre, nous allons découvrir ce qu'est plus précisément le protocole HTTP qui dans certain cas peut s'avérer utile.

Le protocole HTTP (HyperText Transfer Protocol) est le protocole le plus utilisé sur Internet depuis 1990.

- *La version 0.9 était uniquement destinée à transférer des données sur Internet « en particulier des pages Web écrites en HTML ».*
- *La version 1.1 du protocole « la plus utilisée » permet désormais de transférer des messages avec des en-têtes décrivant le contenu du message en utilisant un codage de type MIME (voir plus loin), elle supporte les serveurs HTTP virtuels, la gestion de cache et l'identification, le transfert en pipeline (ou pipelinage) et la négociation de type de contenu « format de données, langue ».*
- *La version 2.0 a été publiée en 2014 et pris en compte désormais par Java EE 8.*

Le but du protocole HTTP est de permettre un transfert de fichiers « essentiellement au format HTML » localisés grâce à URL entre un navigateur « le client » et un serveur Web.

Pour plus d'informations sur le protocole HTTP, merci de visiter l'URL : https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Le type MIME

Le type MIME (Multipurpose Internet Mail Extensions) est un standard qui a été proposé par les laboratoires Bell Communications en 1991 afin d'étendre les possibilités du courrier électronique (mail), c'est-à-dire de permettre d'insérer des documents (images, sons, texte, ...) dans un courrier.

Depuis, le type MIME est utilisé d'une part pour typer les documents attachés à un courrier mais aussi pour typer les documents transférés par le protocole HTTP. Ainsi lors d'une transaction entre un serveur web et un navigateur internet, le serveur web envoie en premier lieu le type MIME du fichier envoyé au navigateur, afin que ce dernier puisse savoir de quelle manière afficher le document.

Un type MIME est constitué de la manière suivante : Content-type: type_mime_principal/sous_type_mime

Pour plus d'informations sur le type MIME, merci de visiter l'URL: https://fr.wikipedia.org/wiki/Type_de_médias

Programme CGI

Principe des pages Web dynamiques

Les ressources accessibles par le protocole HTTP peuvent être des fichiers statiques (pages Web déjà constituées et placées sur le serveur Web) ou des programmes qui renvoie un contenu généré dynamiquement à la demande de l'utilisateur (page Web dynamique). Dans ce cas là, la page Web n'existe pas encore, elle est construite en retour d'une requête de l'utilisateur. Par ailleurs, le contenu de cette page n'est pas prédéterminé, il dépend de la demande de l'utilisateur. On interface ces programmes avec le serveur HTTP en utilisant le standard CGI.

Ce type de programme est utilisé dans de nombreux domaines :

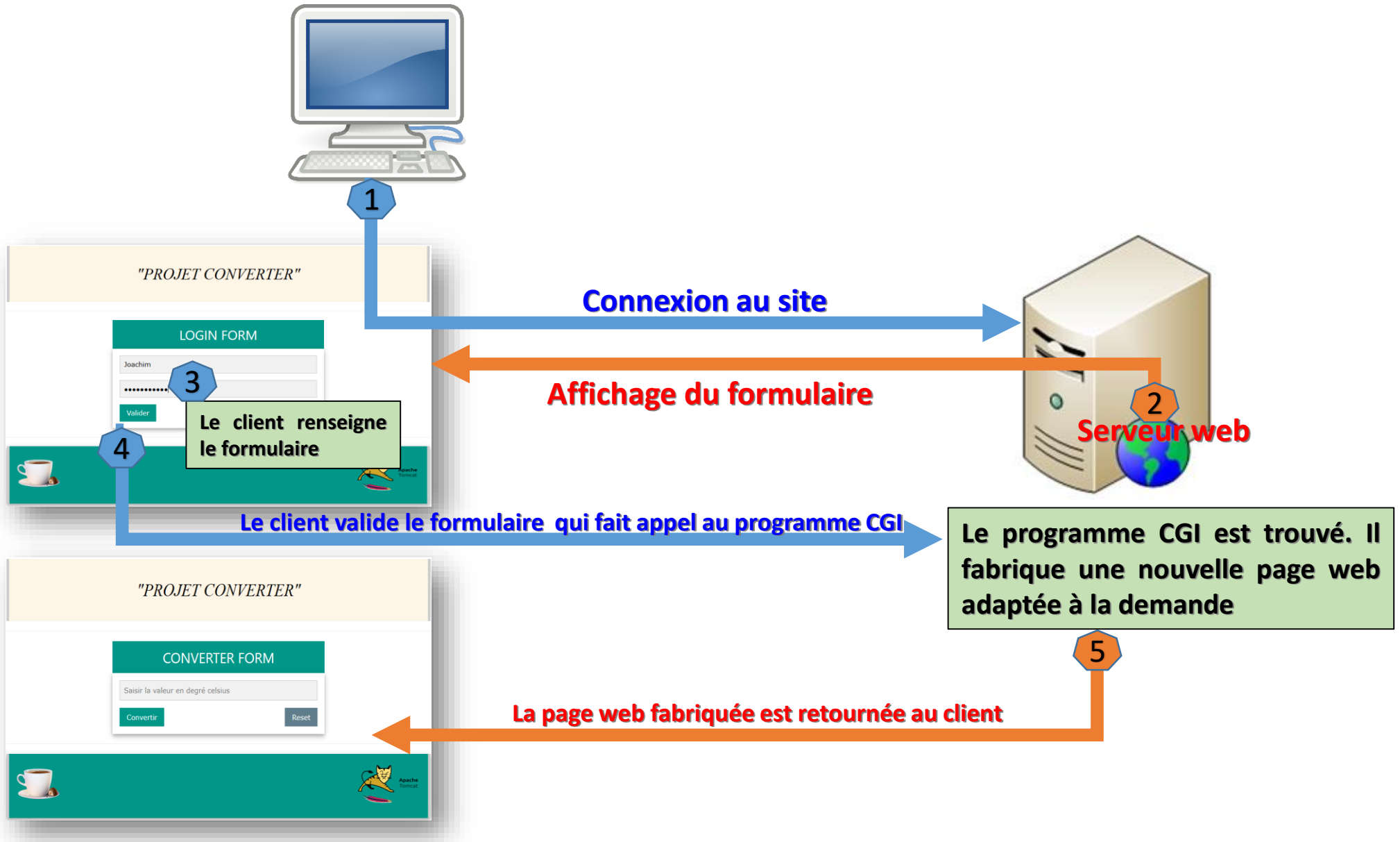
- *moteurs de recherche,*
- *commerce électronique,*
- *gestion des communautés (forums, groupes...).*

Les CGI

Un CGI (Common Gateway Interface, traduisez interface de passerelle commune) est donc un programme exécuté du côté serveur, permettant de cette façon l'affichage de données traitées par le serveur (provenant d'une autre application, comme par exemple un système de gestion de base de données, d'où le nom de passerelle) en réponse à une demande l'utilisateur. C'est l'usage le plus courant des programmes CGI.

Un des grands intérêts de l'utilisation de CGI est la possibilité de fournir des pages dynamiques, c'est-à-dire des pages pouvant être différentes selon un choix ou une saisie de l'utilisateur. L'application la plus fréquente de cette technique repose sur l'utilisation de formulaires HTML permettant à l'utilisateur de choisir ou saisir des données, puis à cliquer sur un bouton de soumission du formulaire, envoyant alors les données du formulaire en paramètre du programme CGI.

Les pages Web dynamiques



Langage de programmation des CGI

Un programme CGI peut être écrit dans n'importe quel langage ou du moins à peu près... pourvu que celui-ci soit :

- *capable de lire le flux de données d'entrée,*
- *capable de traiter des chaînes de caractères,*
- *capable d'écrire sur le flux standard de sortie,*
- *exécutable ou interprétable par le serveur.*

Nous, nous utiliserons le langage Java. Dans ce cas là, deux techniques peuvent être utilisées, soit les servlets, soit les JSP.

Constitution des scripts CGI

Les scripts CGI ont donc pour but d'afficher des pages Web ayant été générées par un programme informatique, d'où la dénomination de pages web dynamiques pour les pages créées par ce moyen. Finalement, les scripts CGI sont tout simplement des scripts de page Web classiques composés des balises « tags » HTML valides. Toutefois, étant donné que le serveur renvoie telles quelles au navigateur les informations que lui fournit le script CGI, il est nécessaire d'ajouter aux données à afficher les en-têtes HTTP permettant au navigateur de comprendre qu'il s'agit d'une page web...

Le programme CGI doit créer lui-même les en-têtes HTTP.

En effet, lorsqu'un programme CGI renvoie un fichier, il doit commencer par envoyer un en-tête HTTP permettant de préciser le type de contenu envoyé au navigateur appelé type MIME, c'est-à-dire:

- *dans le cas d'un fichier html, la chaîne suivante: content-type : text/html*
- *dans le cas par exemple d'un fichier gif « pour faire de la création dynamique d'images par exemple, comme dans le cas d'histogrammes ou de diagrammes divers », la chaîne suivante: content-type : image/gif*

Les pages Web dynamiques

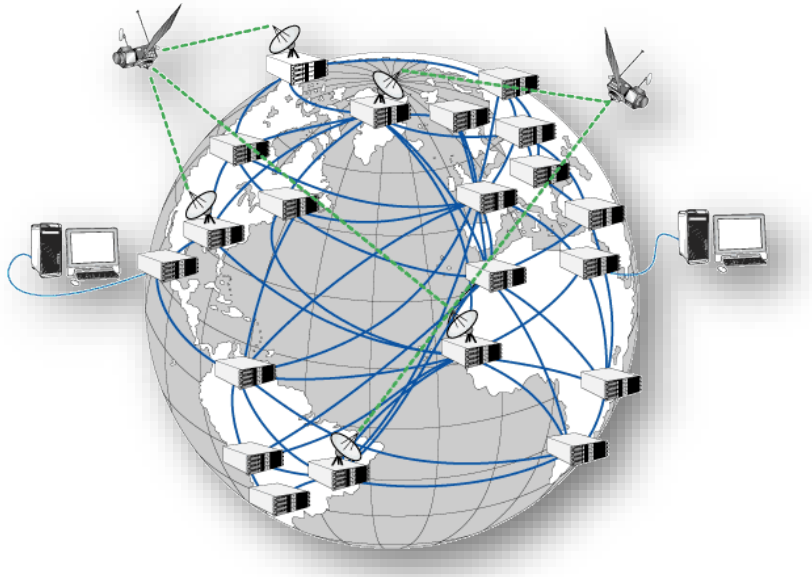
Vous vous demandez sûrement pourquoi le serveur ne pourrait pas ajouter tout seul les en-têtes HTTP, comme il le fait dans le cas des pages web statiques (fichiers .htm et .html). En fait, comme nous venons de le voir, un programme CGI peut renvoyer n'importe quel type de contenu, c'est-à-dire qu'il est capable de renvoyer une image octet par octet, qui sera intégrée dans un document HTML par exemple, pourvu que le CGI renvoie un en-tête correspondant au type de l'image. Une fois de plus, le serveur pourrait éventuellement être capable de reconnaître le type de données que le CGI renvoie et adapter les en-têtes HTTP en fonction. En réalité les en-têtes HTTP peuvent faire beaucoup plus que préciser le type de document envoyé, il est par exemple possible d'effectuer une redirection en renvoyant un en-tête de redirection. Une des applications peut par exemple consister à pointer vers un CGI, qui va enregistrer des informations sur le visiteur (une sorte de compteur de visites amélioré), puis le diriger vers un document...

Comme dit plus haut dans ces lignes, nous utiliserons le langage Java comme programme CGI. La prise en charge des pages web dynamiques est de la responsabilité de la plate-forme Java EE.

Que veut dire java ee ?

Java EE signifie Java Entreprise Edition et représente essentiellement des applications d'entreprise. Cela inclut le stockage sécurisé des informations, ainsi que leur manipulation et leur traitement : factures clients, calculs d'amortissement, réservation de vols, etc.

Ces applications peuvent avoir des interfaces utilisateurs multiples, par exemple une interface Web pour les clients, accessible sur Internet et une interface graphique fonctionnant sur les ordinateurs de l'entreprise sur le réseau privé de celle-ci.



Elles doivent gérer les communications entre systèmes distants, s'occuper automatiquement des différents protocoles de communication, synchroniser les sources avec éventuellement des technologies différentes, et s'assurer que le système respecte en permanences les règles de l'activité de l'entreprise, appelés règles "métier". Pour finir, ces applications s'occupe également automatiquement de la base de données sans que le développeur est à intervenir (bien entendu si le
besoin s'en fait sentir).

Serveurs d'applications

Tout comme les bibliothèques d'interfaces graphiques comme Swing ou JavaFX fournissent les services nécessaires au développement d'applications graphiques, les serveurs d'applications mettent à disposition les fonctionnalités permettant de réaliser des applications d'entreprise : communication entre ordinateurs, mis en place de protocole adaptés, gestion des connexions avec une base de données, présentation de pages Web, gestion des transactions, etc.

Java EE propose justement un ensemble de bibliothèques avec des objets de très haut niveau pour mettre en œuvre facilement ses serveurs d'applications. Chacun de ces objets est adaptée à la situation en correspondant parfaitement au canevas de l'ensemble du processus. Ainsi, les développeurs n'ont pas à partir d'une feuille blanche et surtout Java EE permet d'avoir une démarche standardisée.

Qu'est-ce que java ee ?

Pour de nombreux développeurs, Java EE est souvent synonyme de Entreprise JavaBeans. En fait, Java EE est beaucoup plus que cela. En simplifiant, nous pouvons dire que Java EE est une collection de composants, de conteneurs et de services permettant de créer et de déployer des applications distribuées au sein d'une architecture standardisée.

Java EE est logiquement destiné aux gros systèmes d'entreprise. Les logiciels employés à ce niveau ne fonctionne pas sur un simple PC mais requière une puissance beaucoup plus importante. Pour cette raison, les applications doivent être constituées de plusieurs composants pouvant être déployés sur des plateformes multiples afin de disposer de la puissance de calcul nécessaire. C'est la raison d'être des applications distribuées.

Java EE fournit un ensemble de composants standardisés facilitant le déploiement des applications, des interfaces définissant la façon dont les modules logiciels peuvent être interconnectés, et les services standards, avec leur protocole associé, grâce auxquels ces modules peuvent communiquer.

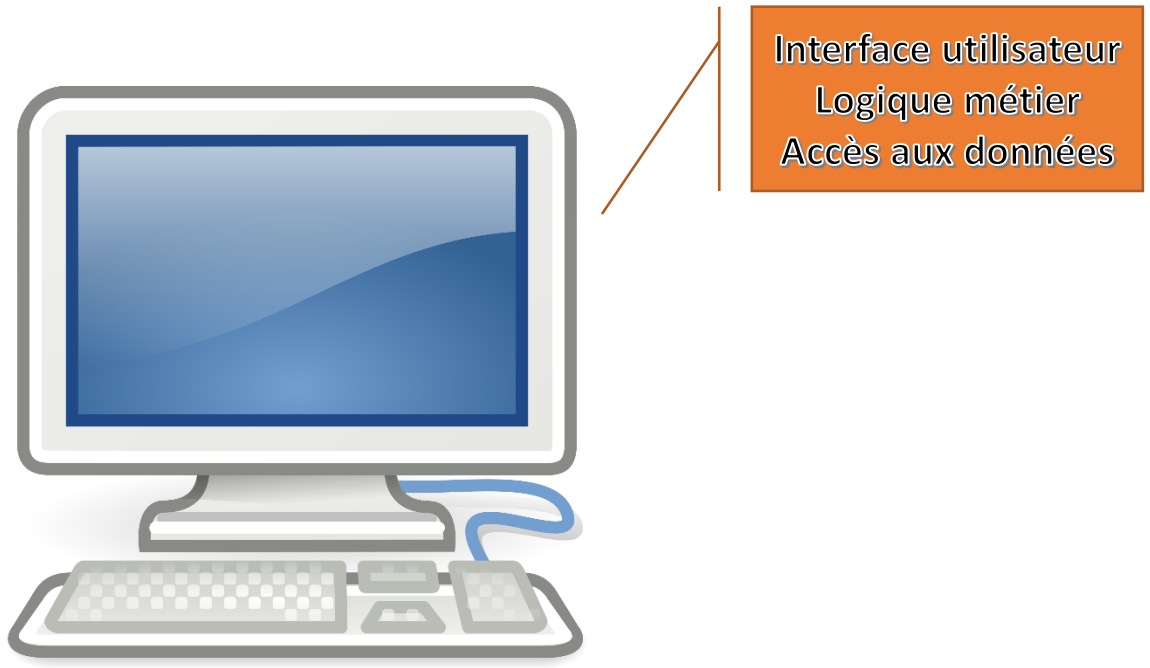
Architecture multi-tiers

Un des thèmes récurrent du développement d'applications Java EE est la décomposition de celles-ci en plusieurs parties ou « tiers ». Généralement, une application d'entreprise est composée de trois couches fondamentales « d'où le terme décomposition en trois tiers » :

1. *La première a pour rôle d'afficher les données pour l'utilisateur et de collecter les informations qu'il saisit. Cette interface est souvent appelée couche de présentation car sa fonction consiste à présenter les données à l'utilisateur et à lui permettre de fournir des informations au système. La couche présentation est la partie de l'application responsable de la création et du contrôle de l'interface présentée à l'utilisateur et de la validation de ses actions.*
2. *Sous cette couche de présentation, on trouve la logique métier qui permet à l'application de fonctionner et de traiter les données. Dans une application de paye, par exemple, la logique métier multiplie les heures travaillées par le salaire horaire pour déterminer combien chaque employé doit toucher. La logique métier est mise en œuvre à partir des règles métier. Cette partie de l'application constitue l'essentiel du tiers médian.*
3. *Toutes les applications d'entreprise ont besoin d'écrire et de lire des données. Cette fonctionnalité est assurée par la couche d'accès de données, également appelée couche de persistance, qui assure la lecture, l'écriture à partir des différentes sources.*

Architecture Simple

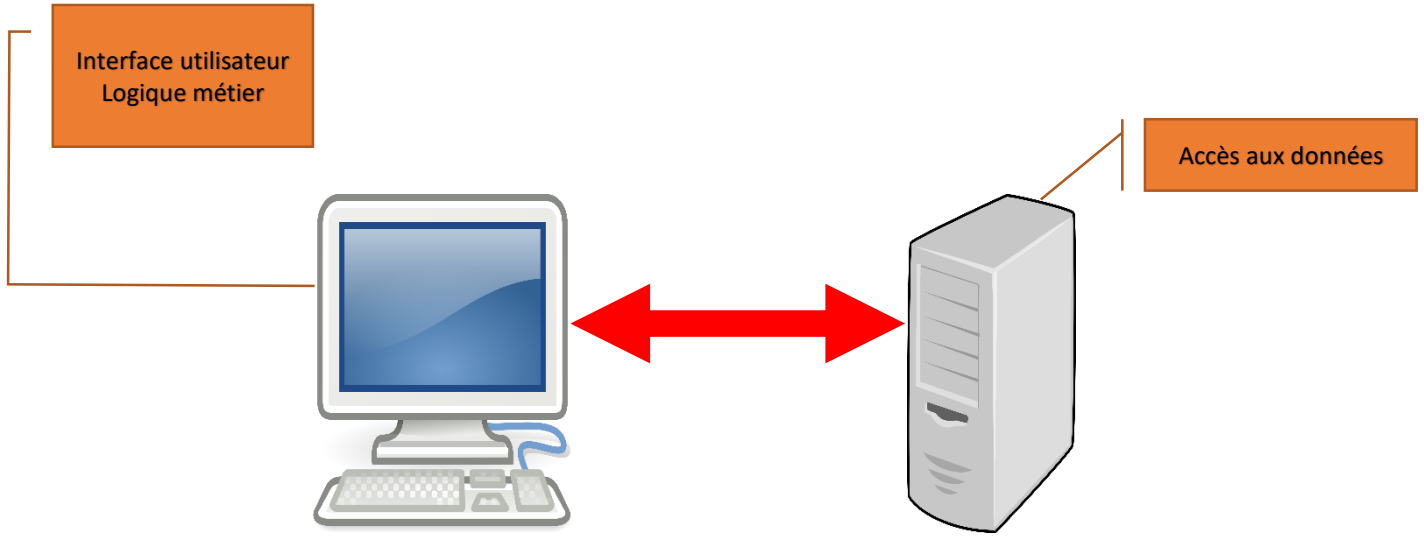
Les applications bureautiques sont conçues pour fonctionner sur un ordinateur unique. Toutes les services fournis par l'application - interface utilisateur, persistance des données (sauvegarde dans des fichiers propriétaires) et logique de traitement de ces données - résident sur la même machine et sont inclus dans l'application. Cette architecture monolithique est appelée simple tiers car toutes les fonctionnalités sont comprises dans une seule couche logicielle.



Architecture Client-Serveur

Les applications plus complexes peuvent tirer parti d'une base de données et accéder aux informations qu'elle contient en envoyant des requêtes à un serveur pour lire et écrire les données. Dans ce cas, la base de données fonctionne dans un processus indépendant de celui de l'application, et parfois sur une machine différente. Les composants permettant l'accès aux données sont séparés du reste de l'application.

La raison de cette approche est de centraliser les données afin de permettre à plusieurs utilisateurs d'y accéder simultanément. Les données peuvent ainsi être partagées entre plusieurs utilisateurs de l'application. Cette architecture est communément appelée client-serveur, qui dans notre approche peut être représentée en deux tiers.



Un des inconvénient de l'architecture deux-tiers est que la logique chargée de la manipulation des données et de l'application des règles métiers afférentes est incluse dans l'application elle-même. Cela pose problème lorsque plusieurs applications doivent partager l'accès à une base de données. Il peut y avoir, par exemple, une règle stipulant qu'un client affichant un retard de paiement de plus de 90 jours verra son compte suspendu. Il n'est pas compliqué d'implémenter cette règle dans chaque application accédant aux données client. Toutefois, si la règle change et qu'un délai de 60 jours est appliqué, il faudra mettre à jour toutes les applications, ce qui peut être contraignant.

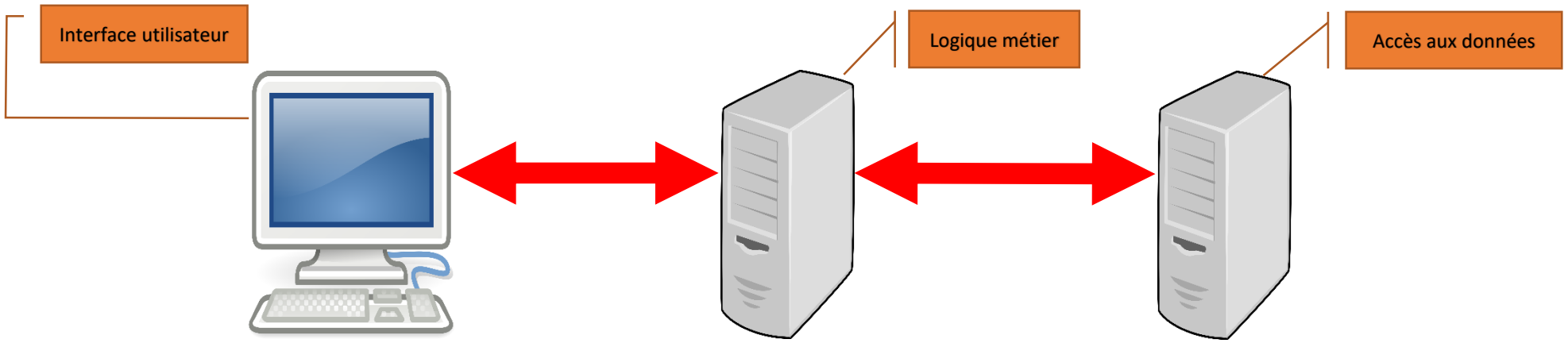
Architecture Trois-Tiers

Pour éviter cette pagaille, la solution consiste à séparer physiquement les règles métier en les plaçant sur un serveur où elles n'auront à être remise à jour qu'une seule fois, et non autant de fois qu'il y a d'applications qui y accède. Cette solution ajoute un troisième tiers à l'architecture client-serveur.

Selon ce modèle, toute la logique métier est extraite de l'application cliente. Celle-ci n'est plus responsable que de la présentation de l'interface à l'utilisateur ou « View » et de la communication avec le tiers médian. Elle n'est plus responsable de l'application des règles. Son rôle est réduit à la couche présentation.

Un des avantages essentiel de cette architecture est qu'elle rend possible la création d'applications dans lesquelles les classes définies au niveau de la logique métier sont directement tirées du domaine de l'application. Le code de cette couche peut utiliser des classes modélisant les objets du monde réel « par exemple des clients » au lieu de manipuler des requêtes SQL complexes.

En plaçant les détails de l'implémentation dans les couches appropriées et en concevant des applications fonctionnant avec des classes modélisant les objets du monde réel, les applications sont plus faciles à maintenir et à faire évoluer.



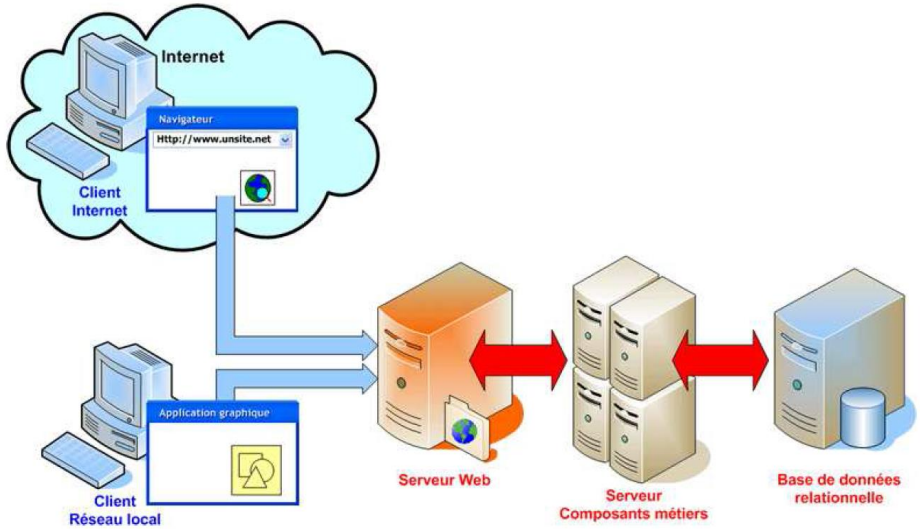
Concepts et spécificités de Java EE

Les termes "client" et "serveur" recouvrent des concepts spécifiques dans le contexte Java EE

Côté Client

Un client Java EE peut être une application console (texte seulement) écrite en Java, ou une application dotée d'une interface graphique développée par exemple en Java FX. Ce type de client est appelé client lourd, en raison de la quantité importante de code qu'il met en œuvre.

Un client Java EE peut également être conçu pour être utilisé à partir du Web. Ce type de client fonctionne à l'intérieur d'un navigateur Web. La plus grande partie du travail est reportée sur le serveur et le client ne comporte que très peu de code. Pour cette raison, on parle de client léger. Un client léger peut être une simple interface HTML, une page contenant des scripts JavaScript, ou encore une applet Java si une interface un peu plus riche est nécessaire.



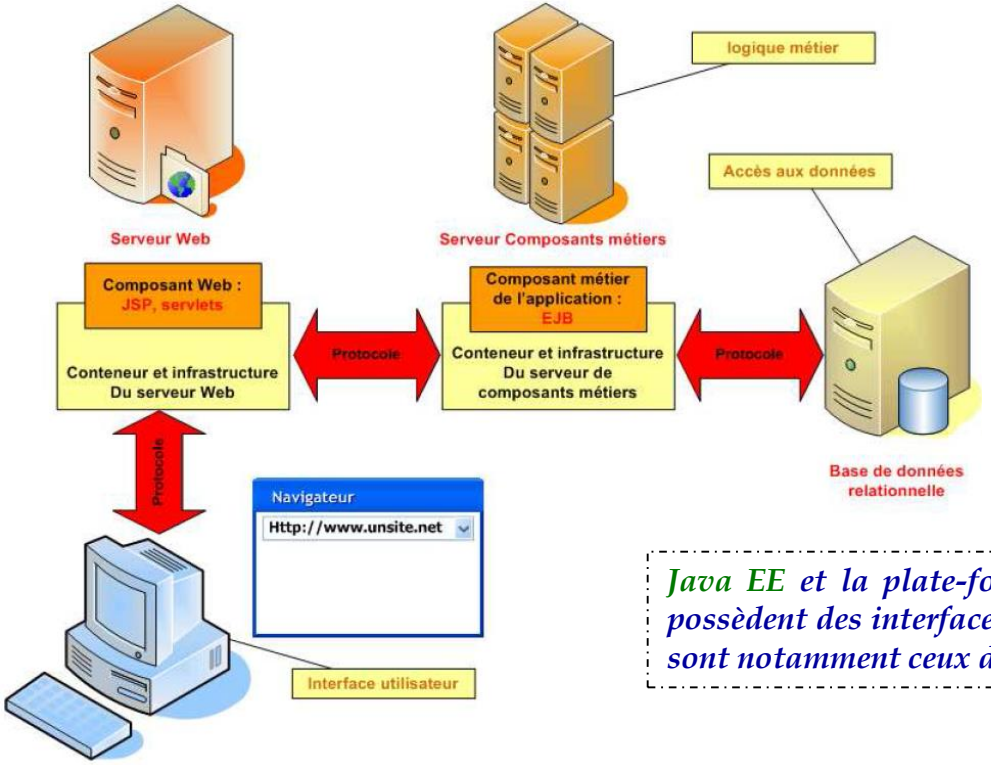
Côté Serveur

Les composants déployés sur le serveur peuvent être classés en deux groupes. Les composants Web sont réalisés à l'aide de servlets ou de JavaServer Pages « JSP ». Les composants métiers, dans le contexte Java EE, sont des Entreprise JavaBeans « EJB ».

Les Conteneurs

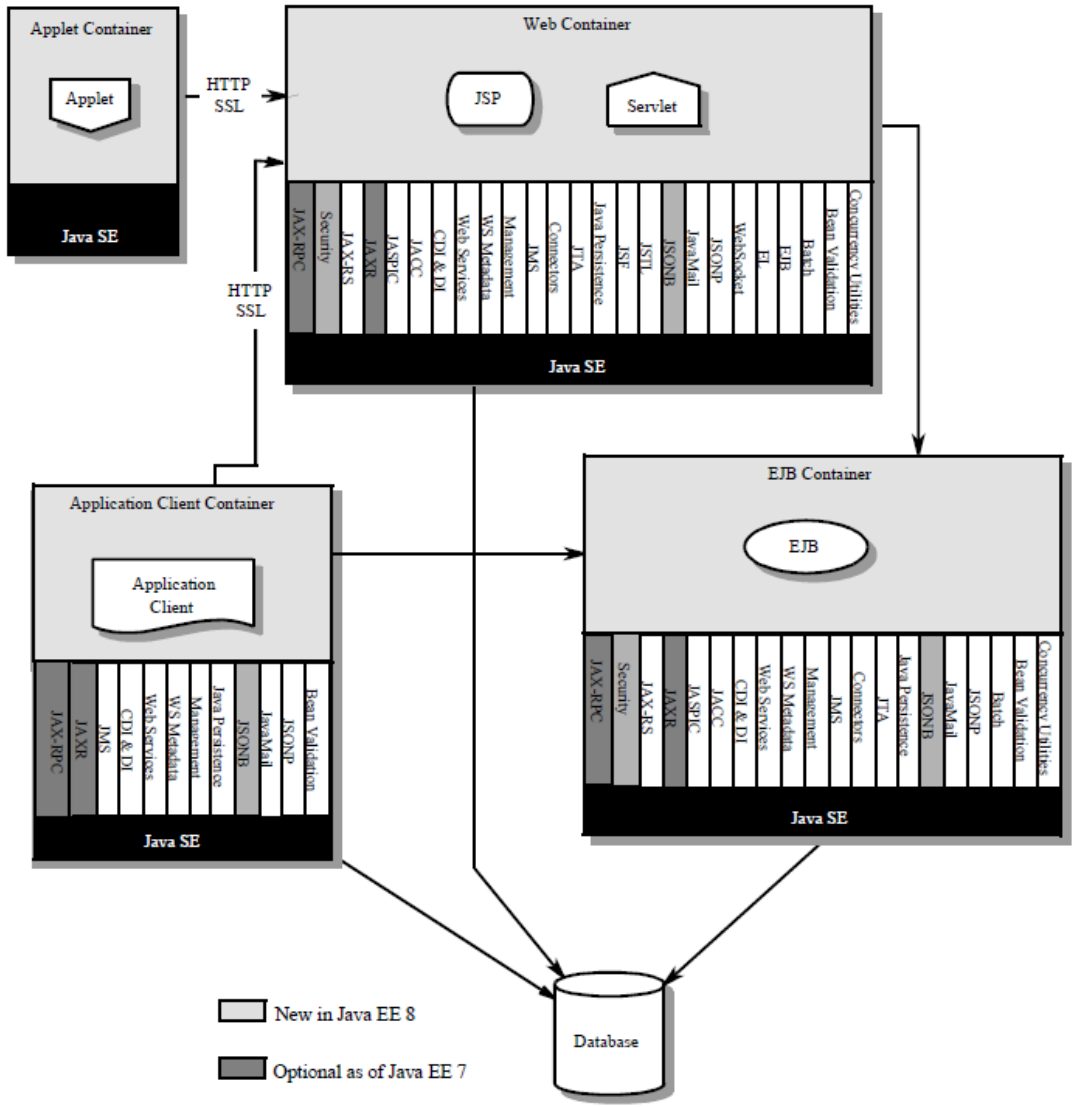
Les conteneurs sont les éléments fondamentaux de l'architecture Java EE. Les conteneurs fournis par Java EE sont de même type. Ils fournissent une interface parfaitement définie ainsi qu'un ensemble de services permettant aux développeurs d'applications de se concentrer sur la logique métier à mettre en œuvre pour résoudre le problème qu'ils ont à traiter, sans qu'ils aient à se préoccuper de toute l'infrastructure interne.

Les conteneurs s'occupent de toutes les tâches fastidieuses liées au démarrage des services sur le serveur, à l'activation de la logique applicative, la gestion des protocoles de communication intrinsèque ainsi qu'à la libération des ressources utilisées, et bien plus encore.

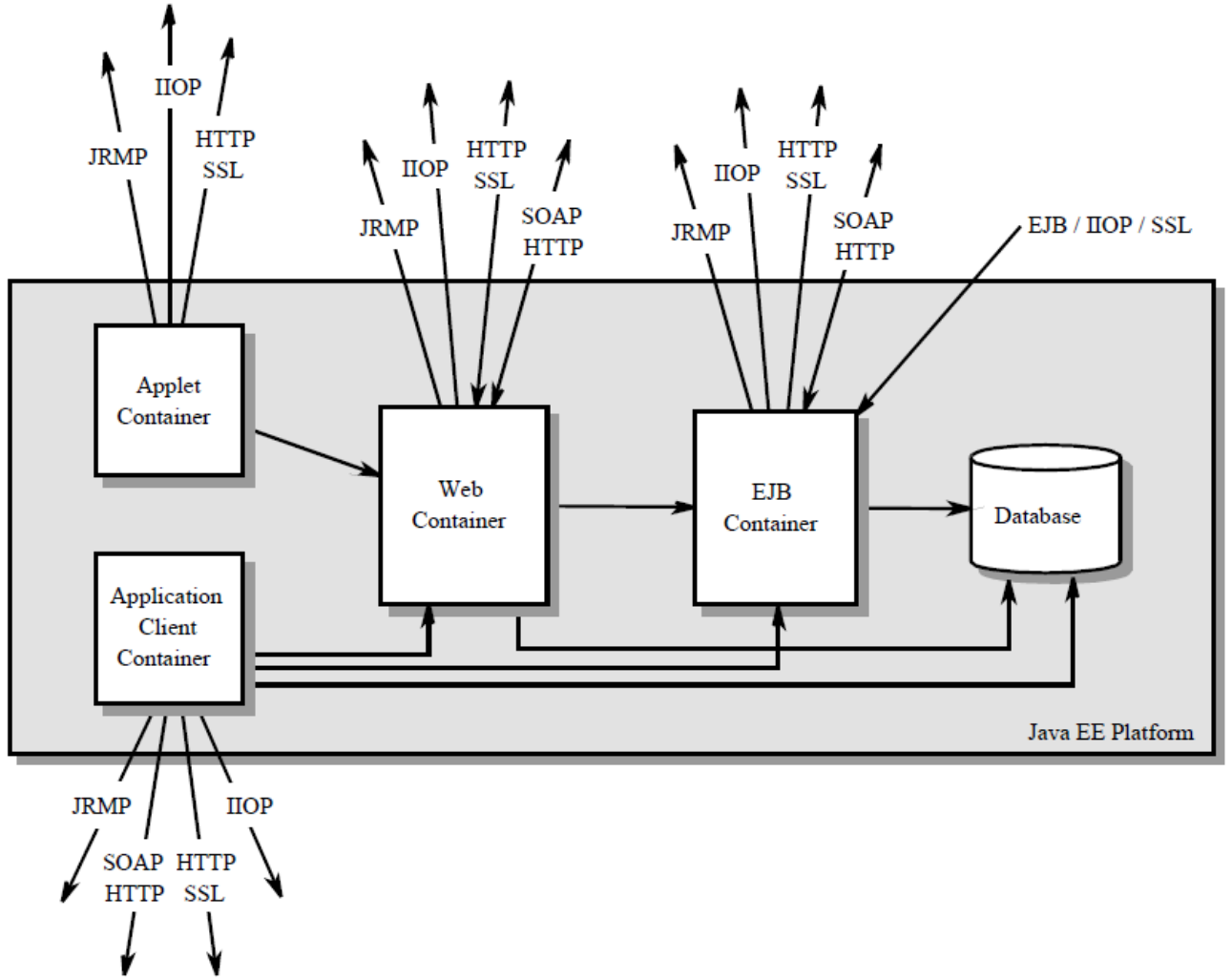


Java EE et la plate-forme Java disposent de conteneurs pour les composants Web et les composants métiers. Ces conteneurs possèdent des interfaces leur permettant de communiquer avec les composants qu'ils hébergent. Les principaux conteneurs Java EE sont notamment ceux dédiées aux EJB, aux JSP, aux servlets et aux clients Java EE.

Les Conteneurs Java EE



L'interopérabilité des conteneurs Java EE



Les Servlets

Introduction

L'étude précédente nous a permis de nous familiariser avec la notion de pages Web dynamiques. Il s'agit maintenant de les construire réellement à l'aide de la technologie Java. Une des technique utilisée consiste à mettre en œuvre des servlets. Les servlets sont des programmes CGI qui sont lancées depuis le serveur Web pour répondre aux différentes requêtes proposées par le client Web.

Ces servlets sont des programmes comme les autres programmes Java mais possèdent en plus des objets spécialisés qui encapsulent entièrement le protocole HTTP. Du coup, la fabrication de page Web à partir du programme CGI que représente la servlet devient très facile à mettre en œuvre. Par ailleurs, ces servlets ne peuvent être lancées qu'à partir d'un serveur Web. Elles ont été spécialement conçues pour cela.

« NB: Les servlets sont conçues comme des extensions de serveurs auxquels s'ajoutent des fonctionnalités. Notez que nous parlons de serveurs et non de serveurs Web. En effet, l'intention originelle était de créer des extensions pour tout types de serveurs, tels les **serveurs FTP ou SMTP.** *Toutefois, seules les servlets HTTP sont aujourd'hui couramment employées ».*

HTTP et les serveurs

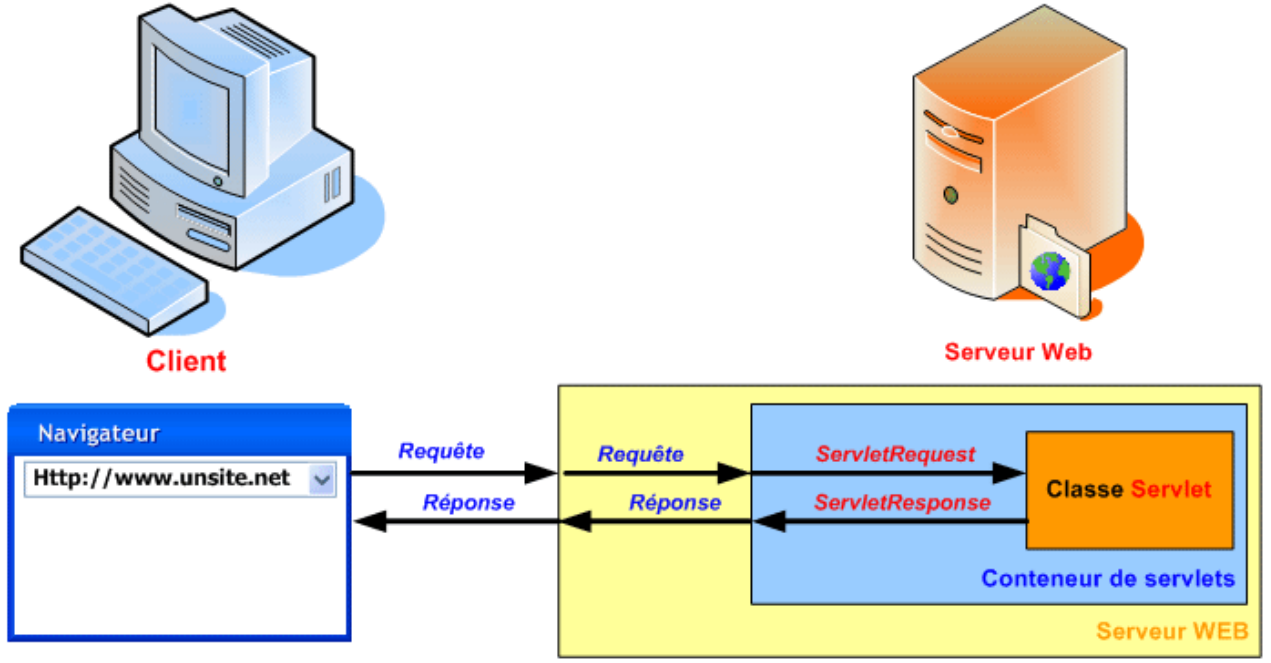
Bien que les servlets aient été conçues originellement pour travailler avec tous les types de serveurs, elles ne sont employées en pratique qu'avec les serveurs Web. Les applications Java EE n'utilisent donc que des servlets répondant à des requêtes HTTP. L'API Servlet contient une classe nommée HttpServlet conçue spécifiquement pour ce type de requêtes. Cette classe est spécialisée dans le traitement du protocole HTTP qui fut développé bien avant l'apparition des servlets et repose sur des fondations particulièrement stables.

Protocole HTTP

Ce protocole définit la structure des requêtes qu'un client peut envoyer à un serveur Web, le format des paramètres pouvant accompagner ces requêtes et la façon dont le serveur doit y répondre. Les servlets HTTP emploient le même protocole pour gérer les requêtes de service et envoyer les réponses aux clients. Il est donc important de bien comprendre les éléments fondamentaux du protocole HTTP pour maîtriser l'utilisation de servlets. *« Pour en savoir plus sur le protocole HTTP, revoyer le chapitre sur les pages Web dynamiques. »*

Le modèle Servlet et les servlets HTTP

Lorsqu'un client « généralement, mais pas nécessairement un navigateur Web » envoie une requête au serveur Web et que celui-ci détermine que la requête est destinée à une servlet, il la passe au conteneur de servlets. Le conteneur de servlets est le programme responsable du chargement, de l'initialisation, de l'appel et de la destruction des instances de servlets.



Les Servlets

Lorsque le conteneur de servlets reçoit la requête, il en analyse l'URI, les entêtes et le corps et stocke toutes les données dans un objet implémentant l'interface javax.servlet.ServletRequest. Il crée également une instance d'un objet implémentant l'interface javax.servlet.ServletResponse. Cet objet encapsule la réponse qui sera envoyée au client. Le conteneur appelle ensuite une méthode de la classe de la servlet, en lui passant les objets requête et réponse.

La servlet traite la requête et renvoie la réponse au client.

Architecture fondamentale d'une servlet

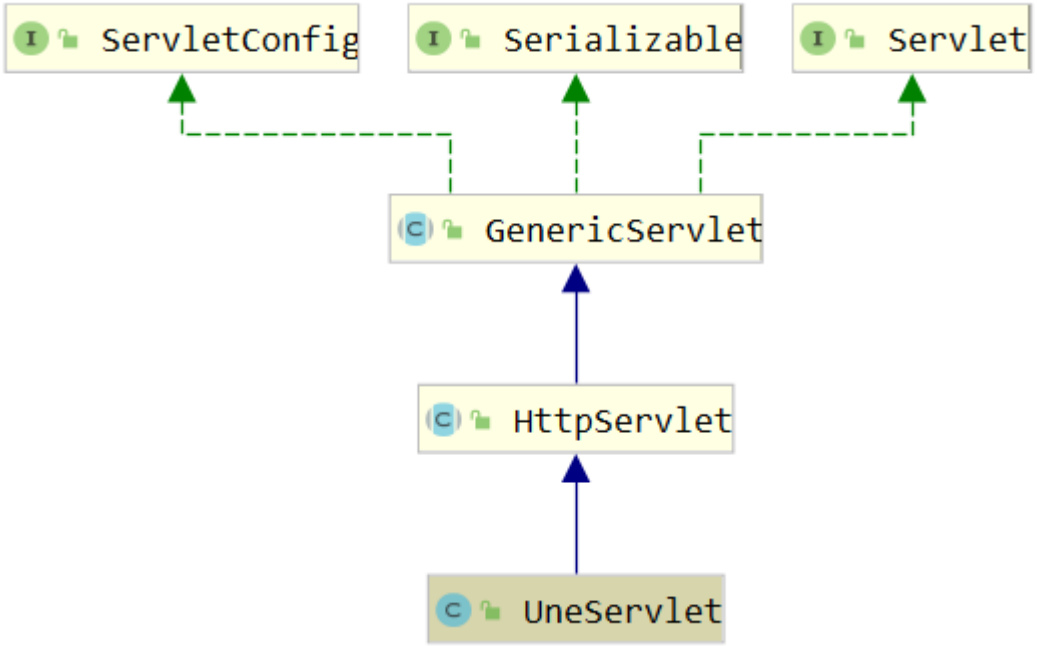
Comme les programmes CGI, les servlets HTTP sont conçues pour répondre aux requêtes GET et POST ainsi qu'aux autres types de requêtes définies par la spécification HTTP. Dans la pratique, vous n'aurez jamais à vous préoccuper d'autres méthodes que GET et POST. Pour écrire une servlet, vous étendrez généralement la classe javax.servlet.http.HttpServlet. Il s'agit de la classe de base de l'API Servlet pour le traitement des requêtes HTTP.

```java
package com.j4ltechnologies.servlets;

import javax.servlet.http.HttpServlet;

/**
 * Classe UneServlet
 * Crée le 28/05/2019 à 18:52
 *
 * @author Joachim Zadi
 */
public class UneServlet extends HttpServlet {

}
```

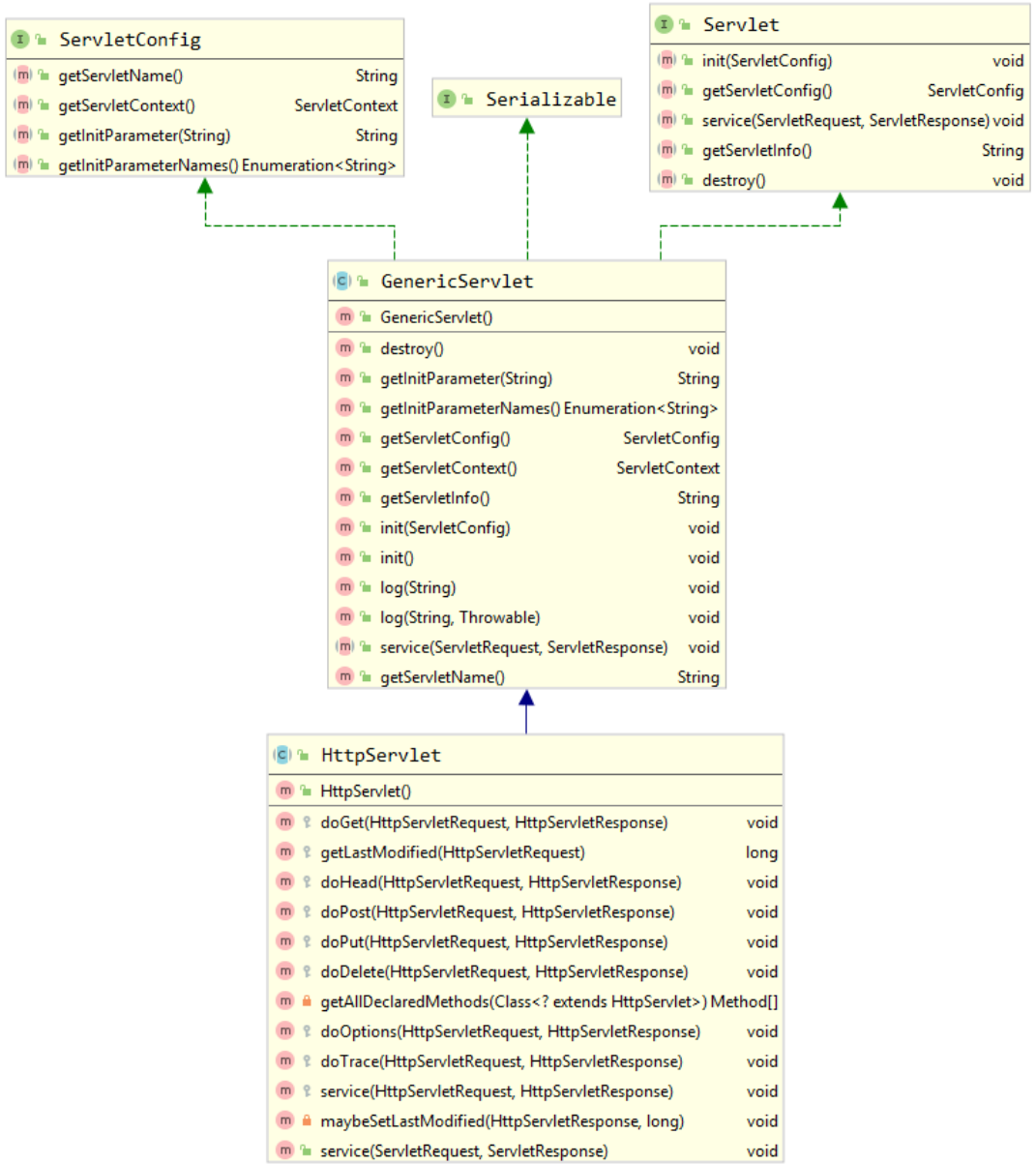
Nous avons crée ici une simple classe java « UneServlet ».
Pour que celle-ci soit une servlet, il faut qu'elle étende la classe « javax.servlet.HttpServlet »,

Cette classe étend elle-même la classe GenericServlet, qui fournit les fonctionnalités fondamentales. Enfin, GenericServlet implémente l'interface principale de l'API, Servlet. Elle implémente également une interface nommée ServletConfig qui offre un accès facile aux données de configuration des servlets.

NB : Les classes et interfaces que nous citons sont celles du package « javax.servlet »,



Les Servlets



La méthode service()

Dans l'interface Servlet, la seule méthode qui gère les requêtes est la méthode service(). Lorsqu'un conteneur reçoit une requête pour une servlet, il appelle systématiquement sa méthode service(). Comme pour toutes les interfaces, une servlet implémentant l'interface Servlet doit obligatoirement fournir une implémentation de toutes les méthodes déclarées, et à fortiori redéfinir la méthode service().

```
public class UneServlet extends HttpServlet {
    @Override
    public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {

    }
}
```

Les méthode doGet() et doPost()

Les classes HttpServlet sont conçues pour réponde aux requêtes HTTP. Elles doivent donc traiter les requêtes GET, POST, HEAD, etc. La classe HttpServlet définit donc des méthodes supplémentaires. La méthode doGet() traite les requêtes GET et la méthode doPost() les requêtes POST. Il existe ainsi autant de méthode doXXX() qu'il y a de type de requêtes HTTP.

Dans cette classe HttpServlet, et donc par défaut, ces méthodes se contentent de retourner au client un message d'erreur indiquant que ce type de requêtes n'est pas supporté comme le montre l'implémentation par défaut de la méthode « doGet() » ci-dessous :

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String protocol = req.getProtocol();
    String msg = lStrings.getString("http.method_get_not_supported");
    if (protocol.endsWith("1.1")) {
        resp.sendError(HttpServletResponse.SC_METHOD_NOT_ALLOWED, msg);
    } else {
        resp.sendError(HttpServletResponse.SC_BAD_REQUEST, msg);
    }
}
```

Les Servlets

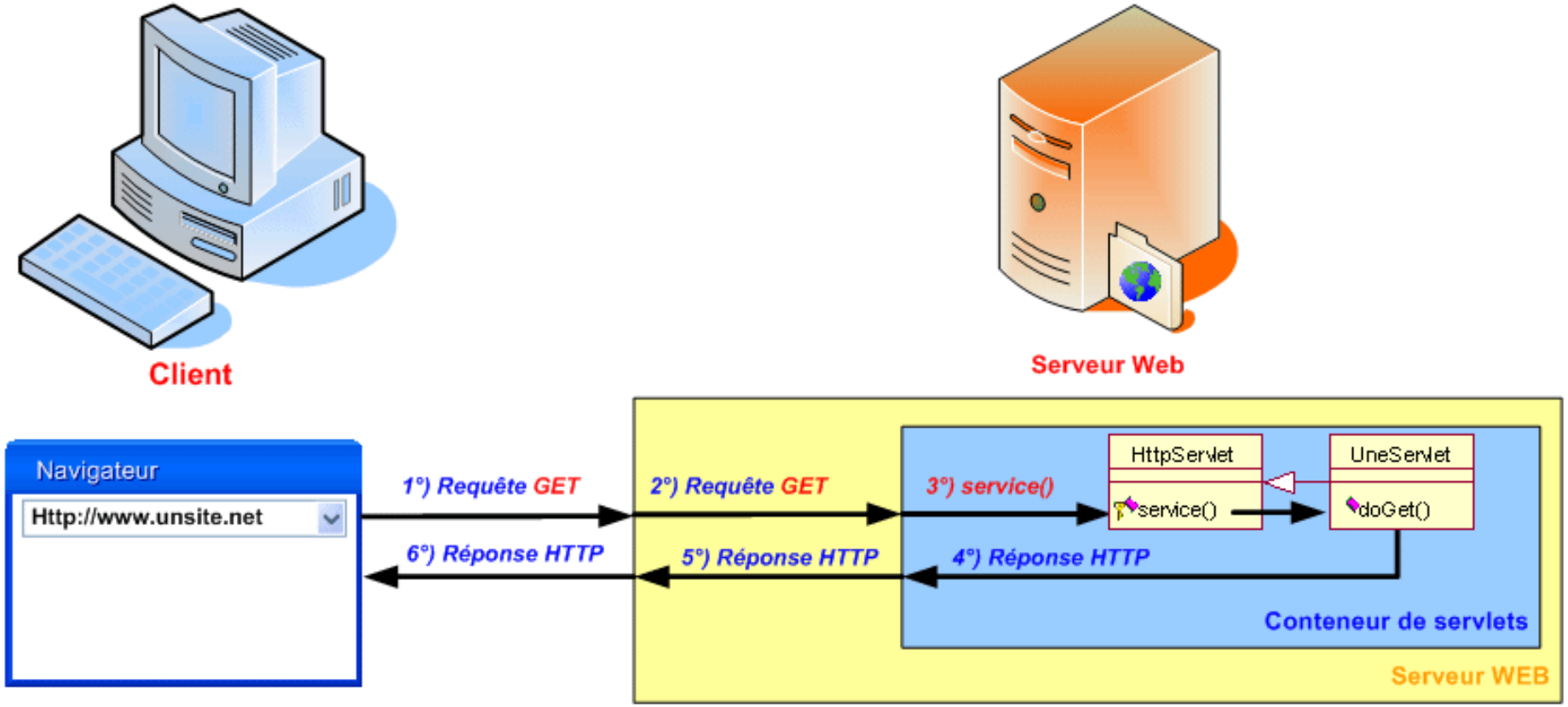
En tant que programmeur, votre rôle consiste à développer une nouvelle servlet adaptée à la situation qui hérite de la classe HttpServlet, et de redéfinir uniquement les méthodes dont vous avez besoin. Le plus souvent, il s'agira de doGet() et de doPost(), comme le montre le listing suivant :

```java
/**
 * Classe UneServlet
 * Crée le 28/05/2019 à 18:52
 *
 * @author Joachim Zadi
 */
public class UneServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        //TODO
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        //TODO
    }
}
```

Certains développeurs étende la classe HttpServlet et redéfinisse uniquement la méthode service() pour traiter tous les types de requête HTTP. Cette solution est acceptable pour un problème simple, mais procéder ainsi peut poser des problèmes graves dans une application Java EE. HttpServlet implémente déjà la méthode service() qui aiguille la requête HTTP vers la méthode adaptée. Il est donc préférable de ne pas le faire et de redéfinir uniquement doGet() et doPost().

Lorsque le conteneur de servlets reçoit les requêtes HTTP, il associe chacune d'elles à une servlet. Il appelle ensuite la méthode service() de celle-ci. En supposant que la servlet étende HttpServlet et ne redéfinisse que les méthode doGet() et doPost(), l'appel à la méthode service() concerne donc celle définit dans la classe HttpServlet. Cette méthode détermine le type de la requête HTTP, puis appelle la méthode doXXX() correspondante. Si votre servlet redéfinit cette méthode, c'est cette implémentation qui sera exécutée. Cette méthode traite la requête, crée la réponse HTTP et la retourne au client.

Les Servlets



Sur cette figure, deux classes ont été représentées : HttpServlet et UneServlet. En fait, il s'agit d'un seul et même objet instance de UneServlet. Cette dernière récupère par héritage la méthode service() issue de HttpServlet.

Les objets request et response

La signature des méthodes doXXX( ) est la suivante :

```
protected void doXXX(HttpServletRequest request, HttpServletResponse response)
```

Chaque méthode « doPost(), doGet(), etc. » prend deux paramètres. L'objet HttpServletRequest encapsule la requête envoyée au serveur. Il contient toutes les données de la requête, ainsi que certains en-têtes. Les méthodes de l'objet request permettent d'accéder à ces données. Les méthodes de l'objet HttpServletResponse encapsule la réponse au client.