Application Web et J2EE

Servlet, JSP, Persistence, Méthodologie

Pierre Gambarotto
<pierre.gambarotto@enseeiht.fr>

Département Informatique et Math appli
ENSEEIHT

Plan

ENSEEIHT

Plan

Connaissances à acquérir

- Application Client/Serveur et HTTP
- Conception d'une application web
- Modèle MVC
- Démarche sur un projet

Côté technique

- J2EE : Servlet, JSP
- Accès bases de données : jdbc
- Eclipse

Plan

Acquisition des connaissances

seance 1 Application Client/Serveur, HTTP, J2EE :Servlet(base)

séance 2 Communication entre ressources web ; J2EE :Servlet(++)

séance 3 HTML et JSP pour interface utilisateur

Méthodologie de conception et mise en pratique

séance 4 Persistence (jdbc), conception BD(base)

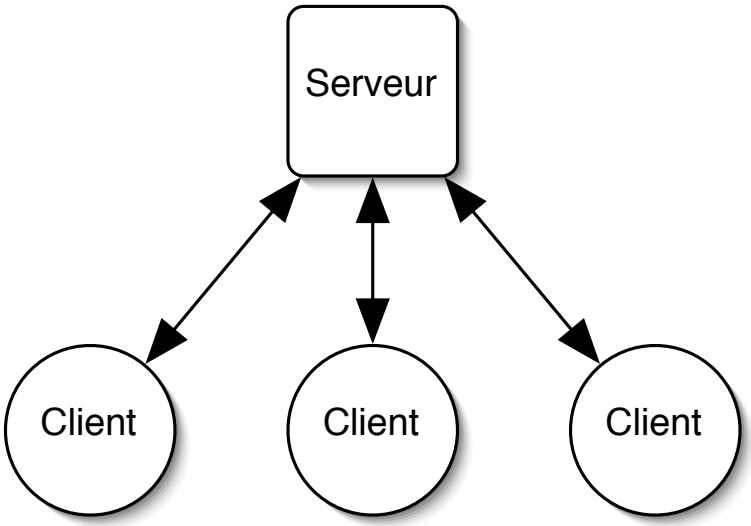
séance 5 MVC

séance 6 Méthodologie de conception

Sommaire

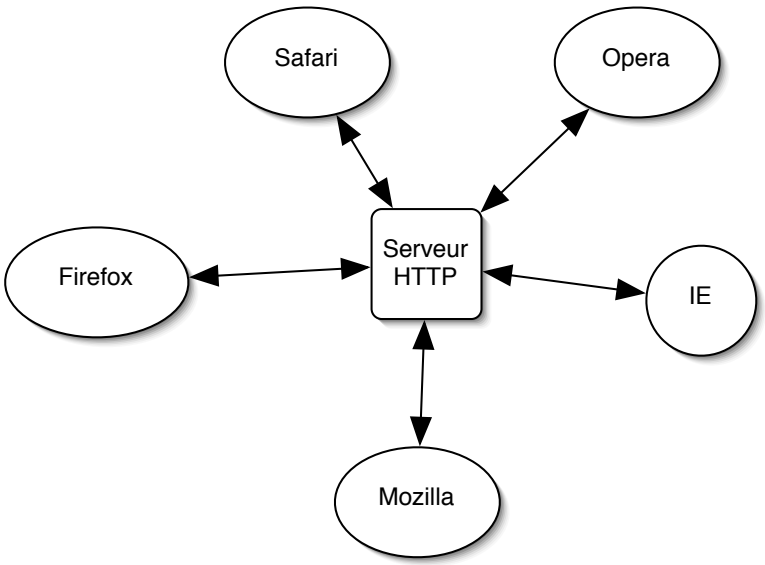
- Application Client Serveur, Application Web
- HTTP
- Web dynamique
- J2EE : présentation
- Servlet : la base

Client/Serveur



- Un Serveur, plusieurs clients
- Client : Interface Utilisateur
- Serveur : partie métier
- Protocole de communication entre le client et le serveur

Application Web

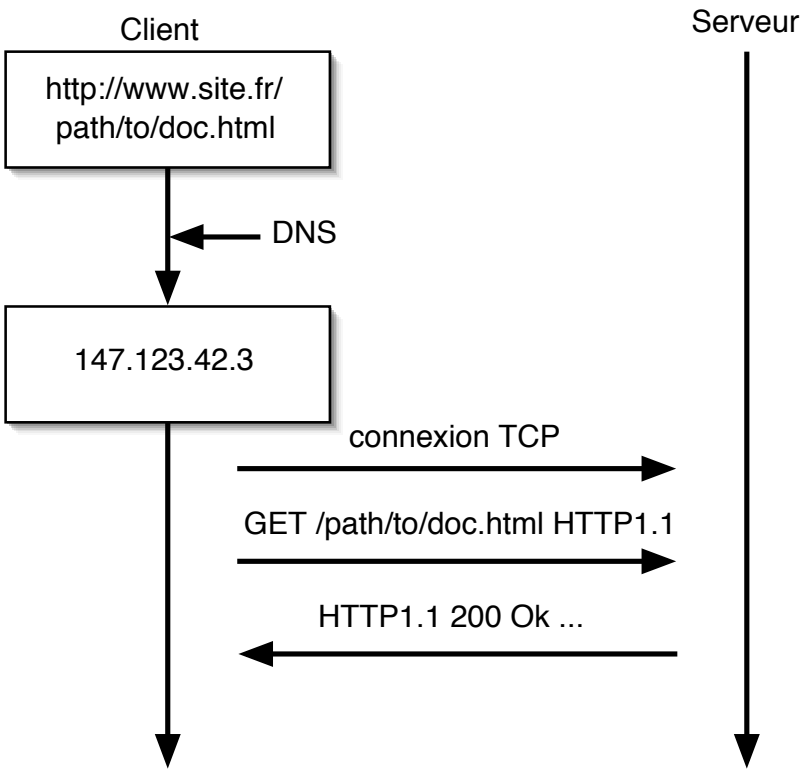


- Client/Serveur
- Client : navigateur web
- Langage de description d'interface : HTML, Javascript
- Serveur : serveur web
- Protocole de communication : HTTP

HTTP en bref

- HyperText Transfer Protocol
- Protocole Client/Serveur, basé sur TCP
- Client=navigateur web
- Désignation de ressource : URL
- HTTP/1.0 : RFC 1945, mai 1996
- HTTP/1.1 : RFC 2616, juin 1999

Exemple de requête



⇒ pas de suivi de requêtes

URL

- Uniform Ressource Locator (URI idem mais Identifier)

```
type_connexion:://serveur/chemin/ressource
   [#id_fragment][?liste_param]
```

- Exemples :

```
http://yuka.enseeiht.fr/svn/cours/j2ee/
http://yuka.enseeiht.fr/doc?param1=val1&p2=val,val2
```

HTTP Requête

1. ligne de commande : requête URI-demandé HTTP-version
2. en-têtes optionnels, sur plusieurs lignes si nécessaire
3. une ligne blanche ;
4. un corps éventuel

Types de reqêtes

- GET : demande l'émission d'une page
- HEAD : demande de lire l'en-tête d'une page
- PUT : demande de mémoriser une page
- POST : demande de traitement du corps de la requête
- DELETE : élimine une page
- LINK/UNLINK : lie/délie deux ressources

En-têtes

- DATE : de la génération de la requête
- REFERER : donne l'URI de la page à partir de laquelle le document est demandé
- USER-AGENT : identifiant du logiciel de navigation employé
- MIME-VERSION : numéro de version
- CONTENT-TYPE : type de données du corps (POST)
- CONTENT-LENGTH : longueur du corps (en bytes)
- CONTENT-ENCODING : codage supplémentaire de la ressource accédée
- CONNECTION : que doit-on faire avec la connection (la garder ouverte ?)
- HOST : indique le nom du serveur (avec éventuellement un numéro de port). Obligatoire en version 1.1

Réponse HTTP

1. une ligne de status : HTTP-version code-réponse phrase réponse
2. en-têtes optionnels
3. une ligne blanche
4. corps éventuel

Code de réponse

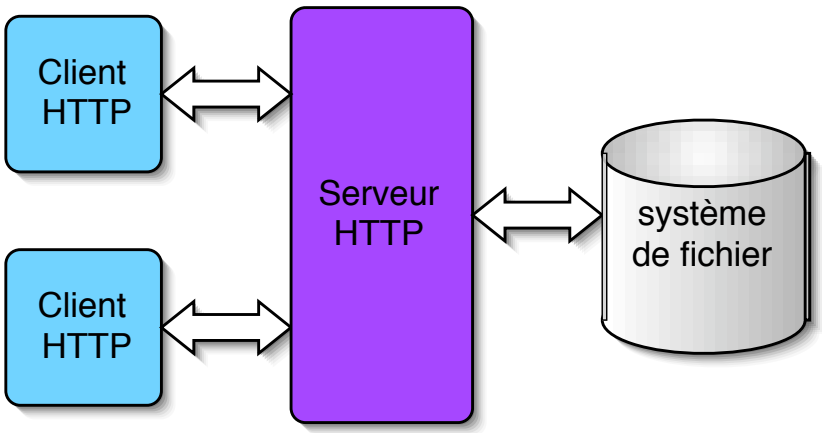
- 1xx : informationnel
- 2xx : succès
- 3xx : redirection
- 4xx : erreur client
- 5xx : erreur serveur

Plan

Mini historique

- crée au CERN pour de la documentation scientifique
- HTTP/1.0 : RFC 1945, mai 1996
- HTTP/1.1 : RFC 2616, juin 1999

Web statique



- ressource demandée : fichier
- serveur web associé à un dossier dans le système de fichiers

Exemple

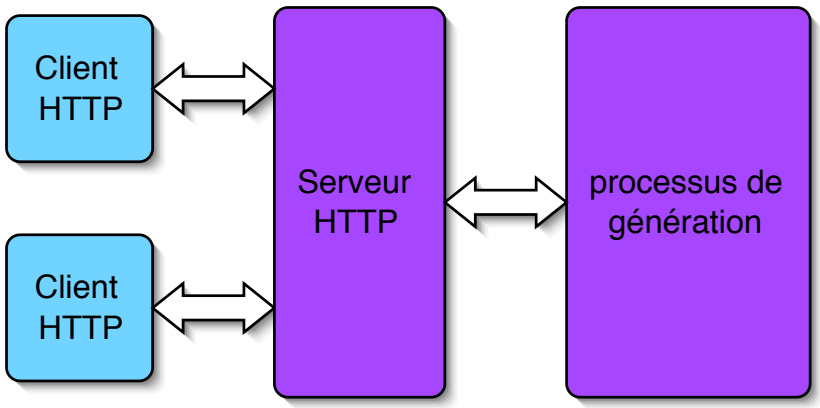
- serveur web `zongo.n7.fr`, associé au dossier `/var/www/html/`
- client :requête de type GET pour l'URL `http://zongo.n7.fr/elfe/dislikeDwarf.html`
- serveur : récupère le fichier `/var/www/html/elfe/dislikeDwarf.html` et l'enrobe dans une réponse HTTP pour l'envoyer au client

Evolution : plus de dynamisme

- nécessité d'une interface utilisateur plus riche : ajout de processus au niveau client pour gérer certains langages : javascript, Applets Java, et autre.
- nécessité de personnaliser les réponses : génération à la voléé de document au niveau serveur

Plan

Génération dynamique de documents



- nom de ressource demandé : document non existant, à générer !
- processus de génération interne ou externe
- Java, php, perl, asp, C#

Exemple

- serveur web `zongodyn.n7.fr`, associé à la configuration : toute les ressources commençant par `/elfe/` sont traitées par le processus `salade.pl`, ici un script perl.
- client :requête de type GET pour l'URL `http://zongo.n7.fr/elfe/dislikeDwarf.html`
- serveur : exécute le processus `salade.pl`, en lui passant la requête reçu en argument, donc en particulier la ressource demandé : `/elfe/dislikeDwarf.html`. Le processus renvoie un résultat, que le serveur enrobe dans une réponse HTTP pour l'envoyer au client

Exemple

- serveur web `zongodyn.n7.fr`, associé à la configuration : toute les ressources commençant par `/elfe/` sont traitées par le processus `salade.pl`, ici un script perl.
- client :requête de type GET pour l'URL `http://zongo.n7.fr/elfe/dislikeDwarf.html`
- serveur : exécute le processus `salade.pl`, en lui passant la requête reçu en argument, donc en particulier la ressource demandé : `/elfe/dislikeDwarf.html`. Le processus renvoie un résultat, que le serveur enrobe dans une réponse HTTP pour l'envoyer au client

Plan

J2EE : collection de produits

- J2EE : Java Edition Entreprise : nom commercial de SUN
- JVM orienté serveur : beaucoup de thread, usage mémoire
- ensemble de libraires : Servlet, JSP, XML, EJB, XML-RPC, RMI, SOAP ...
- Serveur d'application nécessaire pour l'exécution des servlets, des jsp et des ejb
- Nombreux produits libre et de qualité par la fondation Apache.
- ensemble de normes, en particulier : norme pour les serveurs d'application

Servlet

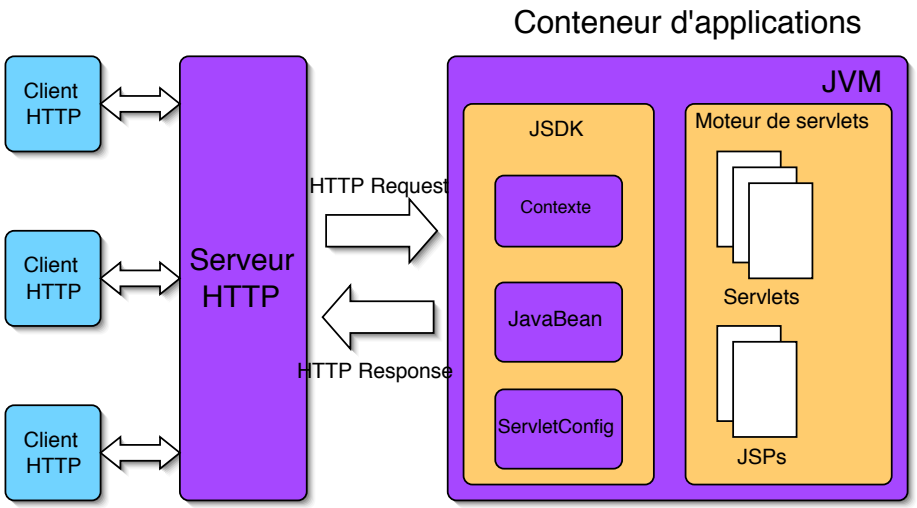
- classe Java implémentant une interface définie dans la librairie `servlet.jar`, correspondant à une norme SUN.
- exécutée par un serveur d'appliation pour générer la réponse à une requête HTTP

JSP

- Java Server Pages
- document texte
- base : langage à balises, HTML ou XML
- introduction de tags dynamiques
- exécutée par un serveur d'application pour générer la réponse à une requête HTTP, et donc idem Servlet

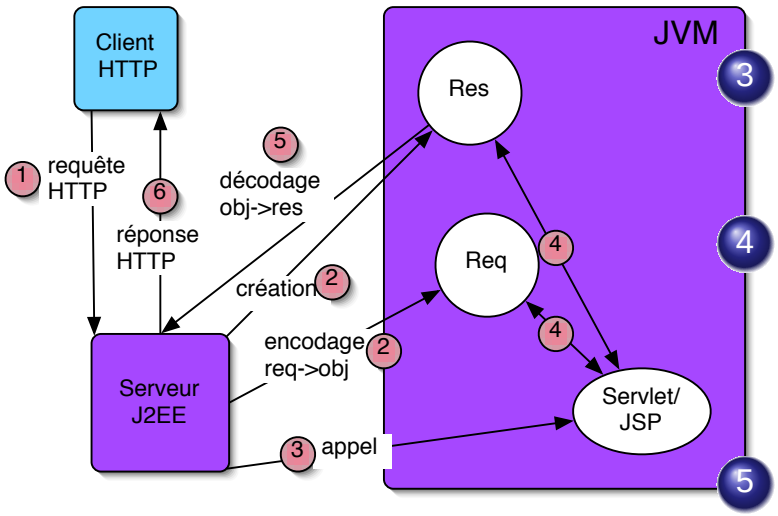
Plan

Web dynamique en Java



- déclinaison en Java d'un serveur HTTP dynamique
- au cœur : une JVM

Principe de traitement des requêtes



1. réception de la requête
2. création de l'objet réponse
   création de l'objet requête
3. appel de la Servlet/JSP associée au nom de ressource demandée
4. la classe associée s'exécute, en accédant aux objets requête et réponse.
5. le serveur décode l'objet réponse, et crée une réponse HTTP
6. la réponse HTTP est renvoyée au client

Application J2EE

- ensemble de fichiers nécessaire à l'exécution
- statique : html, txt, images . . .
- dynamique : servelts et jsp
- librairies utilisées : classes java, libraire .jar
- fichier de configuration : définition du nom utilisé pour appeler les serlets

Serveur J2EE

- plusieurs applications dans des contextes différents
- contexte :
  - nom utilisé dans l'URL pour désigner l'application
  - répertoire local contenant les fichiers de l'application
  - configuration associé à cette application
- sert du contenu statique (fichiers) et dynamique (servlet, jsp)
- configuration générale : nom du serveur, port d'écoute ...

Plan

Organisation des fichiers

```
racine
|-- WEB-INF
|   |-- classes
|   |   |-- Titi.class
|   |   '-- Toto.class
|   |-- lib
|   |   |-- UtilClass.class
|   |   '-- util.jar
|   '-- web.xml
|-- first.jsp
|-- index.html
'-- subrep
    |-- autre.html
    '-- second.jsp
```

- arborescence à racine unique
- présence d'un répertoire `WEB-INF` à la racine
- fichie de configuration : `web.xml`
- classes Java dans le répertoire `WEB-INF/classes`
- librairies dans `WEB-INF/lib`

Ressources d'une applications

- Ressource : nom pouvant être utilisé dans une URL
- tous les noms de fichiers à l'extérieur de `WEB-INF`
- le contenu de `WEB-INF` n'est pas accessible
- association nom/servlet dans `web.xml`

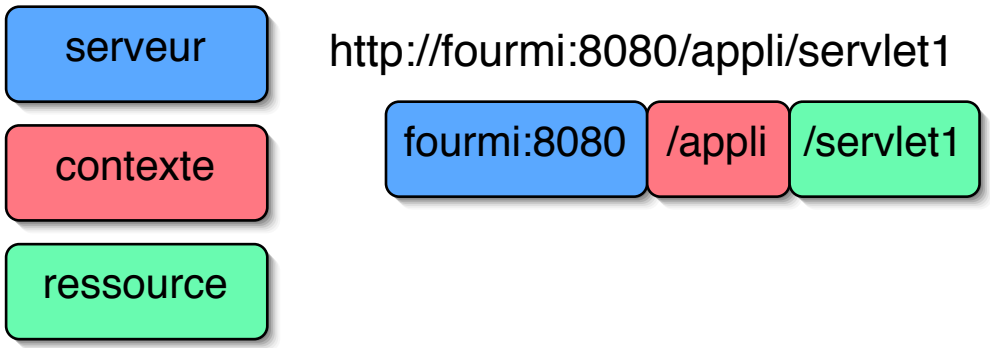
Déploiement et exécution

A faire sur le serveur :

- définition du contexte : nom et emplacement des fichiers
- installation des fichiers de l'application

Exécution à partir d'un client (navigateur web) :

- former l'URL
- regarder le résultat



Exemple : côté développeur

Application fournie par le développeur :

```
|-- WEB-INF
|   |-- classes
|   |   '-- MyClass.class
|   '-- web.xml
'-- index.hml
```

Ressources de cette application :

- ressource statique : `index.html`
- ressource dynamique : `serv1` défini dans `web.xml` comme associé à l'exécution de `MyClass.class`

Côté administrateur du serveur

Configuration choisie par l'adminstrateur du serveur :

- config réseau serveur : @IP, nom DNS, port d'écoute, e.g `zongo:8080`
- nom du contexte :
  `/appli_J2EE/appliPierrot`
- dossier où trouver les fichiers :
  `/home/pierrot/j2ee/zongo`

Exemple : exécution

- URL de base de l'application : `http://zongo:8080/appli_J2EE/appliPierrot`
- exécution de la ressource statique : `http://zongo:8080/appli_J2EE/appliPierrot/index.html` fichier `index.html` récupéré et affiché sur le client
- exécution de la ressource dynamique : `http://zongo:8080/appli_J2EE/appliPierrot/serv1` Le serveur exécute `MyClass.class`, et le résultat est renvoyé au client pour traitement local.
- autre nom de ressource : erreur de type 404, "Document non trouvé".