CONCEPTION ET DEVELOPPEMENT D'APPLICATIONS INTERNET

CDAI 3 - SERVLET

Université Paris Dauphine

Master M2 MIAGE

Année 2014-2015

Bekhouche Abdesslem

Sobral Diogo

DAUPHINE
UNIVERSITÉ PARIS

PLAN

1. INTRODUCTION
2. SERVLET ET ARCHITECTURE N-TIERS
3. MISE EN OEUVRE
4. API
5. EXEMPLE
6. CYCLE DE VIE DES SERVLETS
7. CHAÎNAGE DES SERVLETS
8. COMPLÉMENTS API
9. COOKIES ET SESSION
10. DESCRIPTEUR DE DEPLOIEMENT
11. COMPARATIF JSP/SERVLET
12. CONCLUSION
13. WEBOGRAPHIE

1. INTRODUCTION

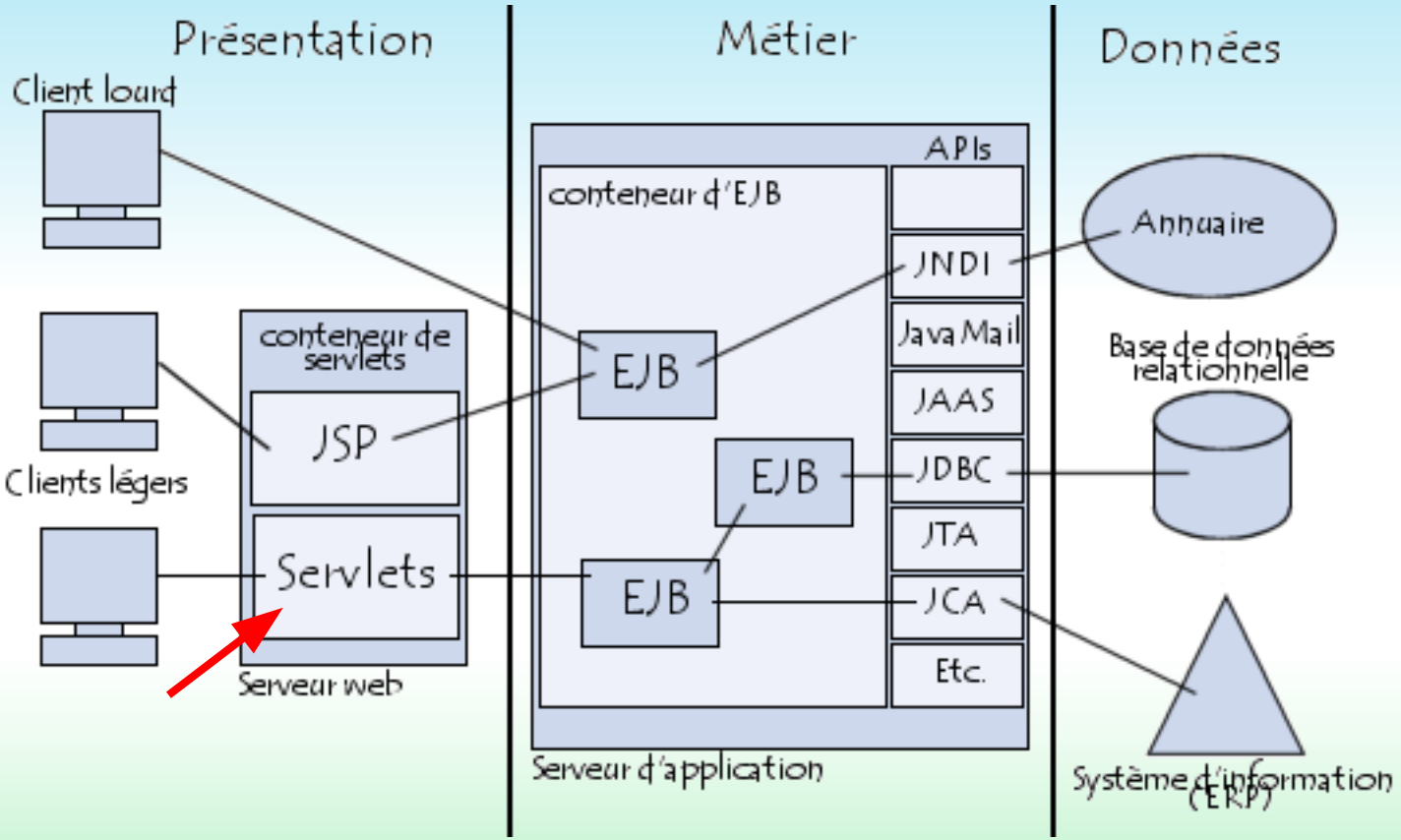
JAVA SERVER PAGES (JSP): HTML + JAVA

SERVLET: JAVA + HTML

- Technologie java qui permet la génération de pages web dynamiques.

- Composant de présentation JEE, comme les JSP:
  - JSP: peu de code java, beaucoup de HTML;
  - Servlet: beaucoup de code java, peu de HTML.

2. SERVLET ET ARCHITECTURE N-TIERS



3. MISE EN OEUVRE 1/2

Écriture d'une servlet = écriture d'une classe Java

Lors du premier chargement d'une servlet (ou après modification), le conteneur Web:

- instancie (alloue et initialise) la servlet;
- servlet = objet Java présent dans le moteur

Puis, ou lors des chargements suivants, le conteneur Web:

- Exécute le code dans un thread;
- Le code produit un résultat qui est envoyé au client;
- En cas d'erreur dans le code Java de la servlet, un message est récupéré dans le navigateur.

3. MISE EN OEUVRE 2/2

Développement d'une Servlet

Utilisation des packages Java javax.servlet.* et javax.servlet.http.*:

- Extension de la classe abstraite *javax.servlet.http.HttpServlet*
- Redéfinition de la méthode doGet ou doPost de cette classe
  - doGet : correspond à une requête HTTP GET
  - doPost : correspond à une requête HTTP POST
- Définit le code à exécuter lorsque la servlet est invoquée automatiquement par le conteneur Web lors d'une requête

void doGet(HttpServletRequest request , HttpServletResponse response);

Requête envoyée par le client: renseigné automatiquement par le conteneur Web.

Réponse HTML retournée par la servlet à renseigner dans le code de la servlet.

4. API 1/2

Méthodes importantes d'un objet request:

- String getParameter(String *param*)
  - Retourne la valeur du champ *param* transmis dans les données du formulaire
- java.util.Enumeration getParameterNames()
  - retourne l'ensemble des noms de paramètres transmis à la servlet
- String getMethod()
  - retourne la méthode HTTP (GET, POST, PUT) utilisée pour invoquer la servlet

4. API 2/2

Méthodes importantes d'un objet response:

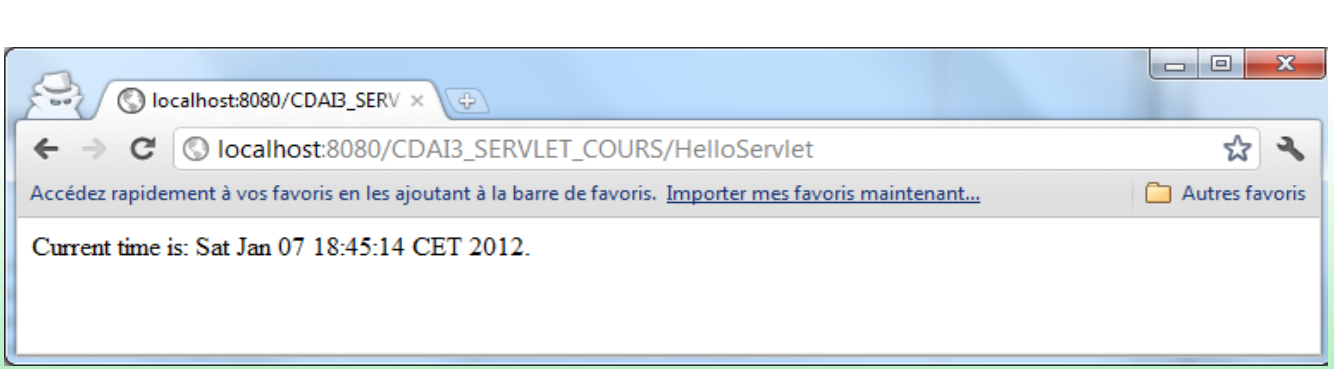
- void setContentType(String type)
  - définit le type MIME du document retourné par la servlet
- PrintWriter getWriter()
  - retourne un flux de sortie permettant à la servlet de produire son résultat
  - la servlet écrit le code HTML sur ce flux de sortie

5. EXEMPLE

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloServlet extends HttpServlet {

 protected void  doGet (HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       try {
         out.println("<html><body>Current time is: " + new Date() + ".</body></html>");
          out.close();
       } finally {
          out.close();
       }
  }
```



6. CYCLE DE VIE DES SERVLETS 1/2

Chaque servlet n'est instanciée qu'une seule fois:

- persistance des variables d'instance entre 2 invocations

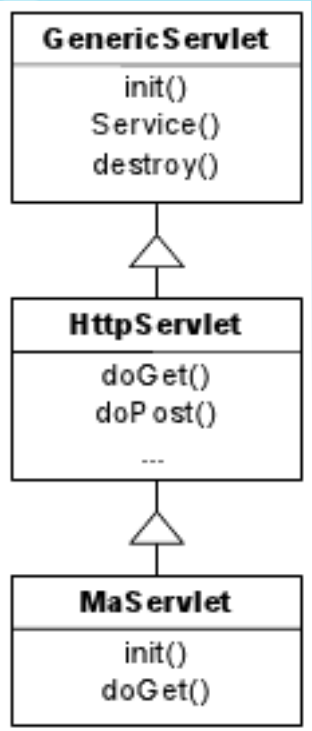
```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloServlet extends HttpServlet {
   int compteur = 0;
protected void  doGet (HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       try {
          out.println( "<h1> "+ compteur++ + "</h1>" );
           out.close();
       } finally {
          out.close();
       }
   }
```

6. CYCLE DE VIE DES SERVLETS 2/2

void init(ServletConfig conf)

- méthode appelée par le moteur au démarrage de la servlet;
- peut être utilisée pour initialiser la servlet;
- propager l'initialisation par super.init(conf);
- ne jamais utiliser de constructeur pour initialiser une Servlet.



void destroy()

- Méthode appelé avant la destruction de la servlet.

Différenciation des méthodes HTTP:

- service() traite toutes les requêtes HTTP;
- doGet(), doHead(), doPost(), doPut() doDelete(), doTrace().

Traitement différencié de chaque requête HTTP

7. CHAINAGE DES SERVLETS

Agrégation des résultats fournis par plusieurs servlets:

- Meilleure modularité;
- Meilleure réutilisation.

Utilisation d'un RequestDispatcher:

- obtenu via l'objet request

```
RequestDispatcher rd = request.getRequestDispatcher("servlet2");
```

Inclusion du résultat d'une autre servlet:

- rd.include(request, response);

Délégation du traitement à une autre servlet:

- rd.forward(request, response);

8. COMPLÉMENTS API 1/2

Méthodes appelables sur un objet request:

- String getProtocol() :
  - retourne le protocole implanté par le serveur (ex. : HTTP/1.1).
- String getServerName() / String getServerPort() :
  - retourne le nom/port de la machine serveur.
- String getRemoteAddr() / String getRemoteHost() :
  - retourne l'adresse/nom de la machine cliente (ayant invoqué la servlet).
- String getScheme() :
  - retourne le protocole utilisé (ex. : http ou https) par le client.
- java.io.BufferedReader getReader() :
  - retourne un flux d'entrée permettant à une servlet chaînée de récupérer le résultat produit par la servlet précédente;
  - permet à la servlet chaînée de modifier le résultat.

8. COMPLÉMENTS API 2/2

Gestion de la concurrence:

- Par défaut les servlets sont exécutées de façon multi-threadée;
- Si une servlet doit être exécutée en exclusion mutuelle (ex. : accès à des ressources partagées critiques):
  - implantation de l'interface marqueur SingleThreadModel

```
public class ConcurrenceServlet extends HttpServlet implements SingleThreadModel{

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {

   /* Tout le code de la Servlet est en exclusion avec lui-même */
   }
}
```

Autre solution : définir du code synchronized dans la servlet

- Déconseillé car le conteneur Web possède des verrous
  Peut conduire à des inter-bloquages

9. COOKIES ET SESSION 1/5

Cookie = donnée stockée par un serveur Web chez un client

- Moyen pour savoir "par où passe" un client, quand, en venant d'où...
- Débat éthique ??
- L'utilisateur a la possibilité d'interdire leur dépôt dans son navigateur

Définis dans la classe javax.servlet.http.Cookie:

- Créé en donnant un nom (String) et une valeur (String) Cookie
  - Cookie uneCookie = new Cookie("sonNom", "saValeur");
- Positionné via un objet response
  - response.addCookie(uneCookie);
- Récupéré via un objet request:
  - Cookie[] desCookies = request.getCookies();
- Quelques méthodes : String getName() / String getValue()

9. COOKIES ET SESSION 2/5

Suivi de session:

- HTTP protocole non connecté
- 2 requêtes successives d'un même client sont indépendantes pour le serveur

Notion de session : suivi de l'activité du client sur plusieurs pages

- Un objet Session associé à toutes les requêtes d'un utilisateur
  (= adresse IP + navigateur)
- Les sessions expirent au bout d'un délai fixé
  (pas de requête pendant n secondes ! expiration de la session)

Consultées/crées à partir d'un objet request:

- HttpSession session = request.getSession(true);
  retourne la session courante pour cet utilisateur ou une nouvelle session
- HttpSession session = request.getSession(false);
  retourne la session courante pour cet utilisateur ou null

9. COOKIES ET SESSION 3/5

Méthodes appelables sur un objet de type HttpSession:

- void setAttribute(String name, Object value);
  - ajoute un couple (name, value) pour cette session
- Object getAttribute(String name);
  - retourne l'objet associé à la clé name ou null
- void removeAttribute(String name);
  - enlève le couple de clé name
- java.util.Enumeration getAttributeNames();
  - retourne tous les noms d'attributs associés à la session
- void setMaxInactiveInterval(int seconds);
  - spécifie le temps avant la fermeture d'une session
- long getCreationTime(); / long getLastAccessedTime();
  - retourne la date de création / de dernier accès de la session en ms depuis le 1/1/1970, 00h00 GMT new Date(long);

9. COOKIES ET SESSION 4/5

Partage de données entre servlets

Contexte d'exécution = ensemble de couples (name, value) partagées par toutes les servlets instanciées

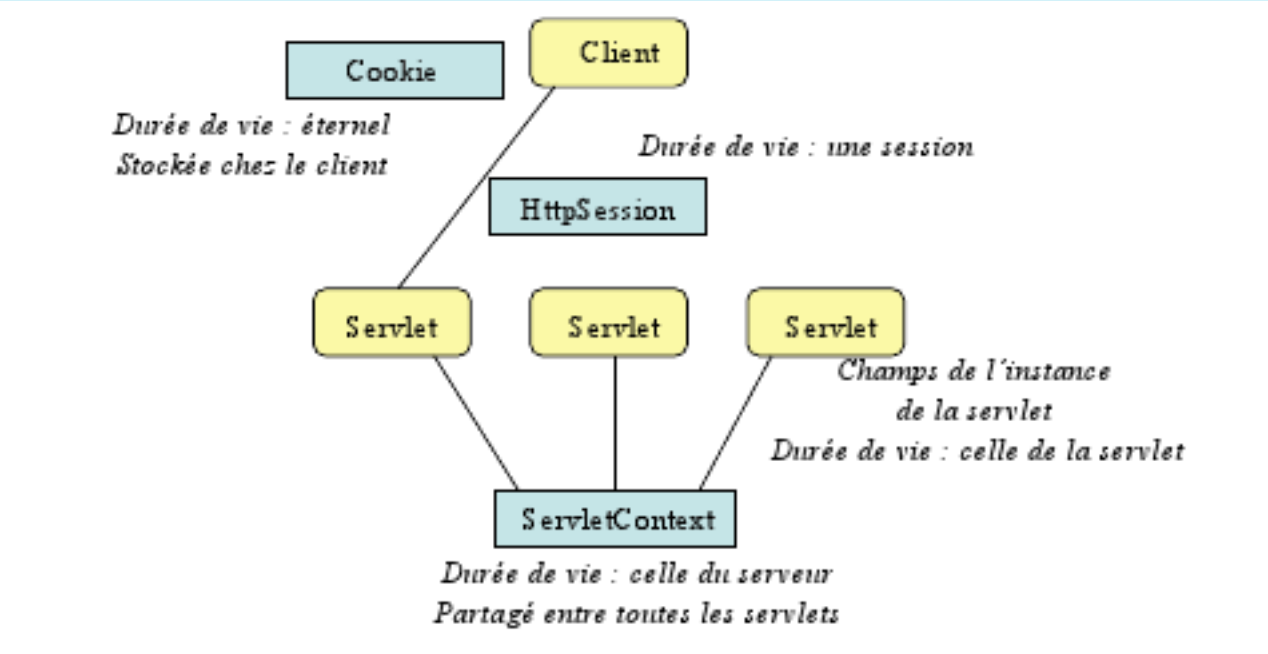
ServletContext ctx = getServletContext()

Méthodes appelables sur un objet de type ServletContext

- void setAttribute(String name, Object value)
  - ajoute un couple (name, value) dans le contexte
- Object getAttribute(String name)
  - retourne l'objet associé à la clé name ou null
- void removeAttribute(String name)
  - enlève le couple de clé name
- java.util.Enumeration getAttributeNames()
  - retourne tous les noms d'attributs associés au contexte

9. COOKIES ET SESSION 5/5

Récapitulatif des objets de données



10. DESCRIPTEUR DE DEPLOIEMENT

- Fichier web.xml servant à décrire les caractéristiques des servlets (voir en TP).

11. COMPARATIF JSP/SERVLET

- JSP compilé en servlet
- servlet : possibilité de distinguer les requêtes HTTP (doPut, doGet, doPost, ...)

- JSP : beaucoup HTML, peu Java
- servlet : beaucoup Java, peu HTML

- contenu autre que HTML (PDF, GIF, Excel, ...) : servlet oui / JSP oui mais
- session, chaînage, redirection : oui dans les 2 cas : API vs directives

- servlet : pur Java : facilement éditable IDE
- JSP : plutôt éditeur de pages HTML

- servlet compilation avant déploiement / JSP après
- JSP à redéployer si erreur

12. CONCLUSION 1/2

- Servlet et Java Server Pages :
  - Permettent d'étendre le comportement des serveurs Web avec des programmes Java

- Résumé des fonctionnalités
  - Code embarqué dans un fichier HTML
  - Portabilité, facilité d'écriture (Java)
  - Notion de session au dessus d'HTTP
  - Persistance des données entre deux appels
  - Pas de persistance si serveur redémarre
  - JSP chargée et instanciée une seule fois
  - JSP exécutée dans des processus légers (threads)

12. CONCLUSION 2/2

Servlets : étendent le comportement des serveurs Web avec des programme Java:

- Portabilité, facilité d'écriture (Java);
- Définition du code, du paquetage, du déploiement;
- Persistance des données dans les servlets;
  Servlet chargée et instanciée une seule fois
- Exécutée en // avec des processus légers (threads).

Mais :
Difficile d'écrire du code HTML dans du code Java
D'où Java Server Pages (JSP)

!!Pas de mécanisme intégré de distribution
Introduction de la technologie Enterprise Java Beans (EJB)

13. WEBOGRAPHIE

http://docs.oracle.com/javaee/7/api/
http://www.servletworld.com/
http://docs.oracle.com/javaee/7/tutorial/doc/
http://www.ai.univ-paris8.fr/~maa/M1_J2EE_Servlet.pdf