Applications Web

Cours 2:
Introduction J2EE
Servlets et JSP

Khaled Khelif

Rappel

- Web statique vs. Web dynamique
- Principe des applications web
- Protocole HTTP : requêtes en mode texte

Développement d'applications Web dynamiques : Utilisation de J2EE

Plan

- J2EE
- Servlets
- JSP

J2EE en bref

- Java 2 Enterprise Edition
- Plate-forme fortement orientée serveur pour le développement et l'exécution d'applications distribuées et donc en particulier Web.

J2EE en bref

2 grands types d'outils:

- Composants web et métiers
- Services d'infrastructure (exp: JDBC, JNDI…) et de communication (exp: JAAS, WS…)

J2EE en bref

Les composants:

- Web
  - partie présentation (interface utilisateur et traitements).
  - Le client reçoit seulement du texte HTML=> la partie visible de l'application.
  - Derrière, différentes technologies => code plus performant, plus robuste, et plus facile à maintenir.
    ⇨ Servlet et JSP

- Métiers
  - composants spécifiques pour le traitement des données et l'interfaçage avec les bases de données.

    ⇨ EJB (Entreprise JavaBean) , JavaBean

J2EE en bref

Les serveurs d'applications J2EE:

- Pour le développement Servlet/JSP ⇨ Conteneur de Servlet
  Tomcat, Resin, Jetty….

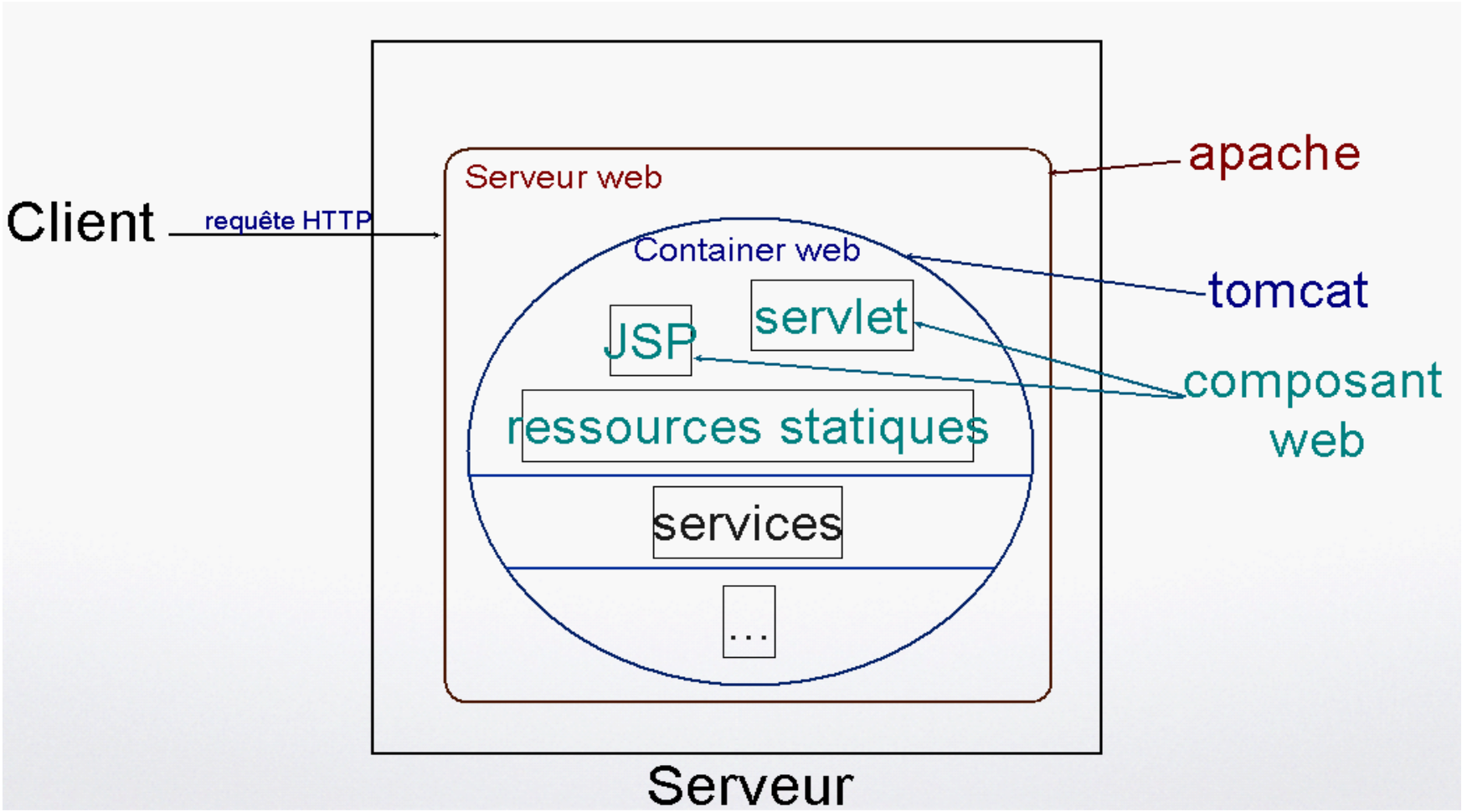
- Pour l'ensemble des spécifications J2EE ⇨ Conteneur d'EJB
  JBoss, Jonas, WebSphere Application Server…

J2EE en bref

Conteneur ou moteur de Servlet

- Mode Autonome
  - Contient également un serveur web
  - Toutes les requêtes passent par le moteur de Servlet
- Mode lié au serveur Web
  - Sollicité uniquement pour le traitement des Servlet

J2EE en bref


Client
requête HTTP
Serveur web
Container web
JSP
servlet
ressources statiques
services
...
Serveur
apache
tomcat
composant
web

J2EE vs .Net

- Un concurrent de taille : Microsoft .net
- Un ensemble de technos rendant les applications facilement accessible par Internet

.net	36%
J2EE commercial	27%
J2EE libre	37%

Servlet: présentation

- Pour la création d'applications dynamiques
- Classe java : chargée dynamiquement, elle étend les fonctionnalités d'un serveur web et répond à des requêtes dynamiquement

  ⇨ Permet de gérer des requêtes HTTP et de fournir au client une réponse HTTP
- S'exécute par l'intermédiaire d'une JVM
- S'exécute dans un moteur de Servlet ou conteneur de Servlet permettant d'établir le lien entre la Servlet et le serveur Web

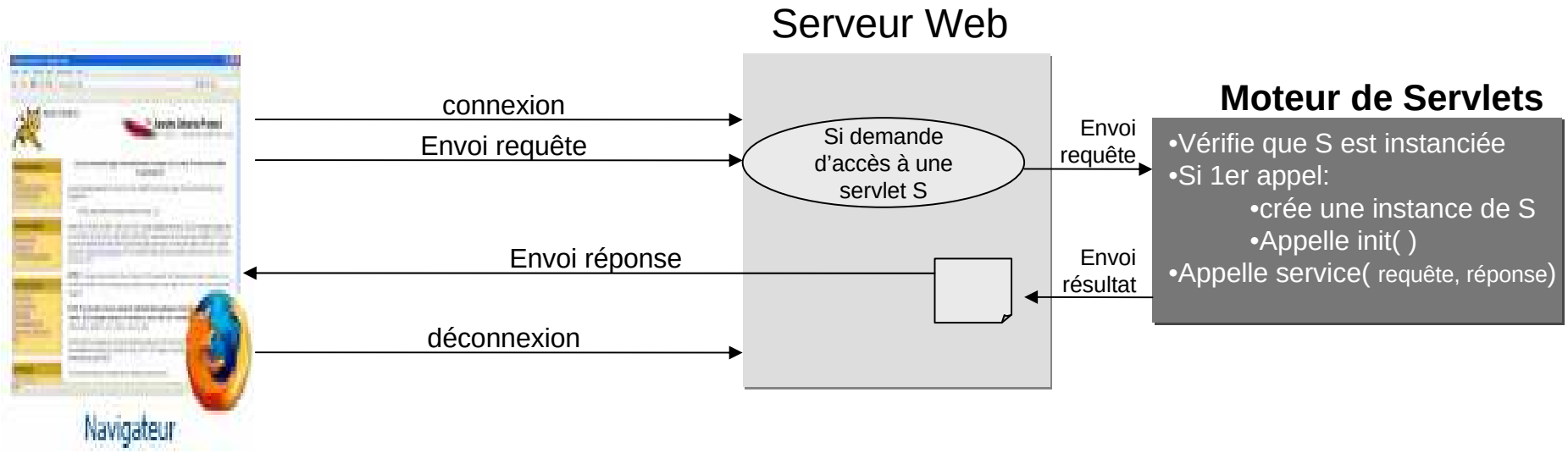
Servlet: avantages

- **Portabilité** :niveau systèmes et serveurs
- **Efficacité**: semi compilée, multithread, gestion du cache, connexions persistantes
- **Puissance**: communication bidirectionnelle avec le serveur web, partage de données entre servlets, chaînage de servlets
- **Pratique**: gestion des cookies, suivi des sessions, manipulation simple du protocole HTTP

Servlet: inconvénient

Limitée en matière de GUI car elle s’exécute cote serveur : couplage avec des technos comme Flash ou les applets

Servlet : fonctionnement



Servlet : cycle de vie

- Une seule instance par Servlet est utilisée
- Une requête client a pour résultat un nouveau thread transmis à service()
- Une Servlet est initialisée, utilisée puis détruite ⇨Cycle de vie de la Servlet

Servlet : cycle de vie

- Initialisation: méthode init()
  - Appelée uniquement lors du 1er appel à la Servlet
  - Effectue des opérations de paramétrage et d'initialisation de la Servlet
  - Avec ou sans paramètres (besoin d'accéder ou non aux paramètres de configuration du serveur)

Servlet : cycle de vie

- Utilisation: méthode service()
  - Appelée pour chaque requête reçue
  - Paramètres: ServletRequest et ServletResponse

- Destruction: méthode destroy()
  - Suppression de l'instance de la Servlet (demande administrateur, temps d'inactivité trop grand)
  - Fermeture de connexion à une BD, fermeture de fichiers…

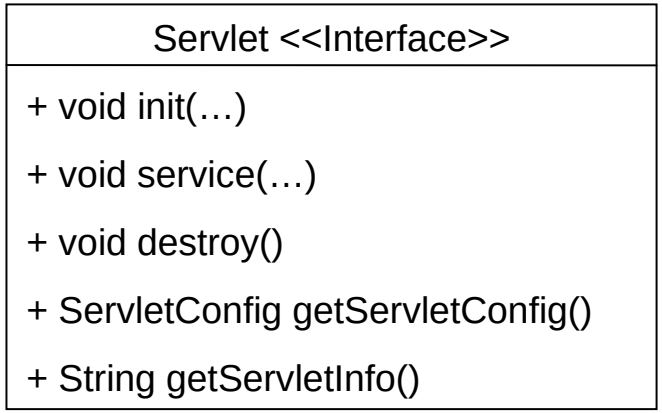
Servlet : cycle de vie

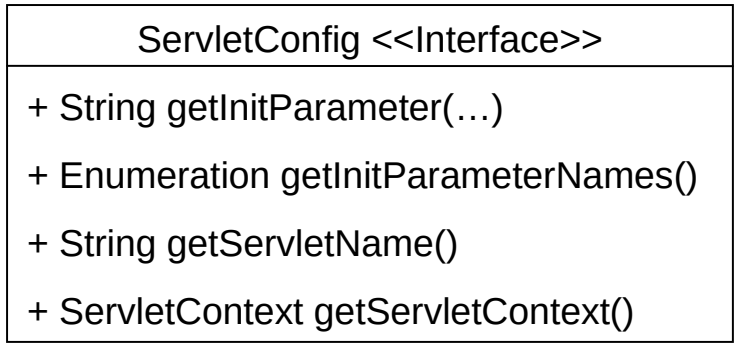
```
public void init() throws ServletException {
    FileReader fileReader = null;
    BufferedReader bufferedReader = null;
    try {
        fileReader = new FileReader("InitCounter.initial");
        bufferedReader = new BufferedReader(fileReader);
        String initial = bufferedReader.readLine();
        count = Integer.parseInt(initial);
    } catch (IOException e) {
        ...
    } finally {
        if (bufferedReader != null)
            bufferedReader.close();
    }
}
```

```
public void destroy() {
    FileWriter fileWriter = null; PrintWriter printWriter = null;
    try {
        fileWriter = new FileWriter("InitCounter.initial");
        printWriter = new PrintWriter(fileWriter);
        printWriter.println(count);
    } catch (IOException e) {
        ...
    } finally {
        if (printWriter != null) printWriter.close();
    }
}
```

Servlet : du générique au http

• Une Servlet doit implémenter l'interface javax.servlet.Servlet et javax.servlet.ServletConfig





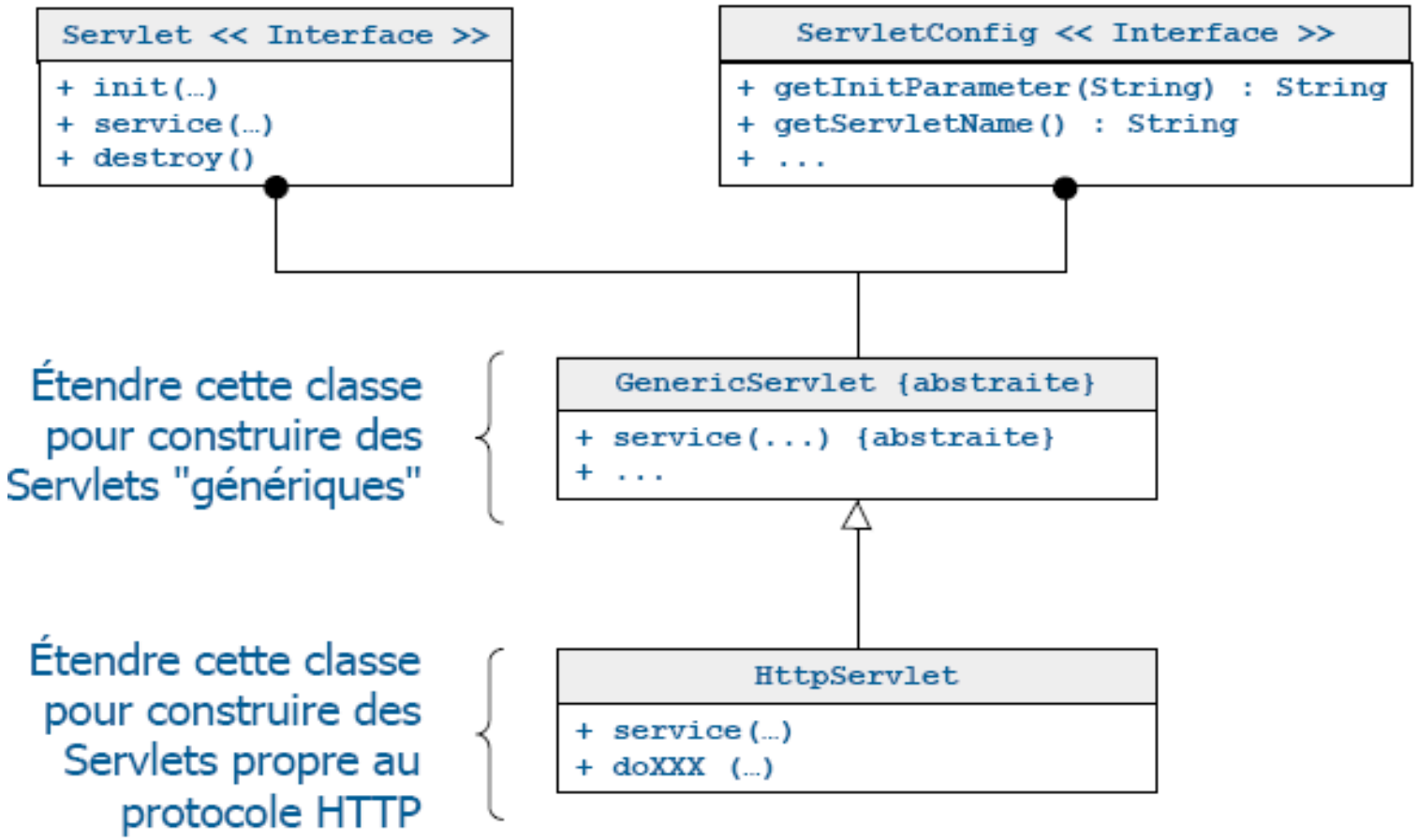
Servlet : du générique au http

• l'API Servlet fournit deux classes qui proposent déjà une implémentation:

  • GenericServlet : pour la conception de Servlets indépendantes du protocole

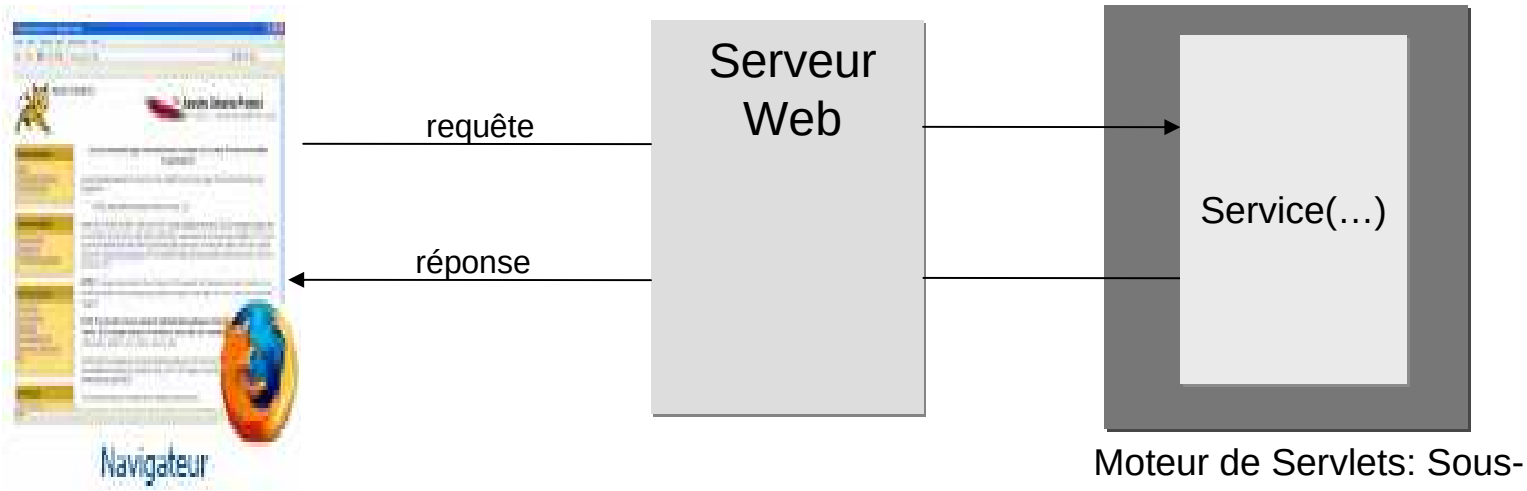
  • HttpServlet : pour la conception de Servlets spécifiques au protocole HTTP

Servlet : du générique au http

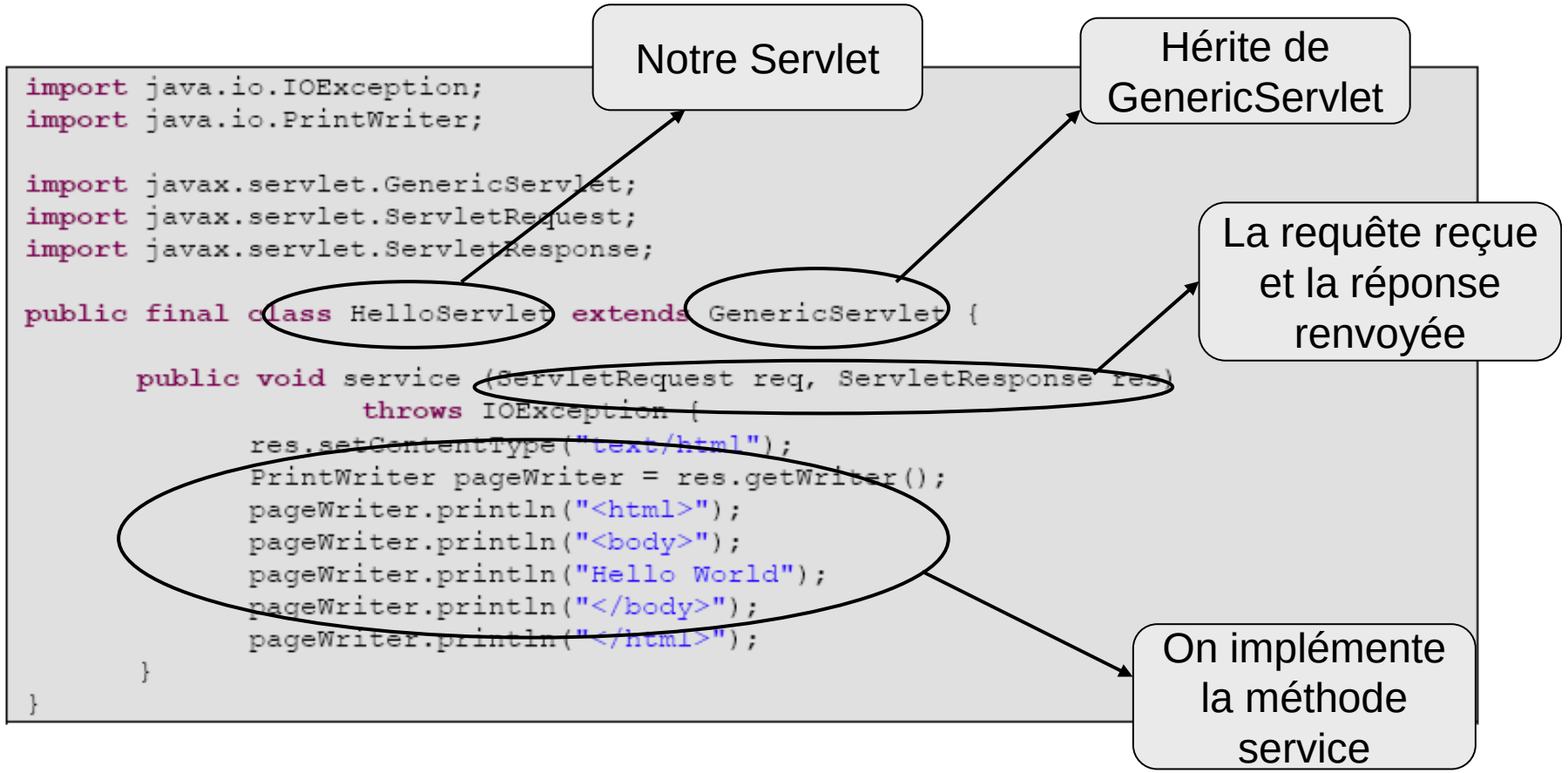


Servlet: GenericServlet

- Indépendante du protocole
- Implémentation de la méthode service(…)

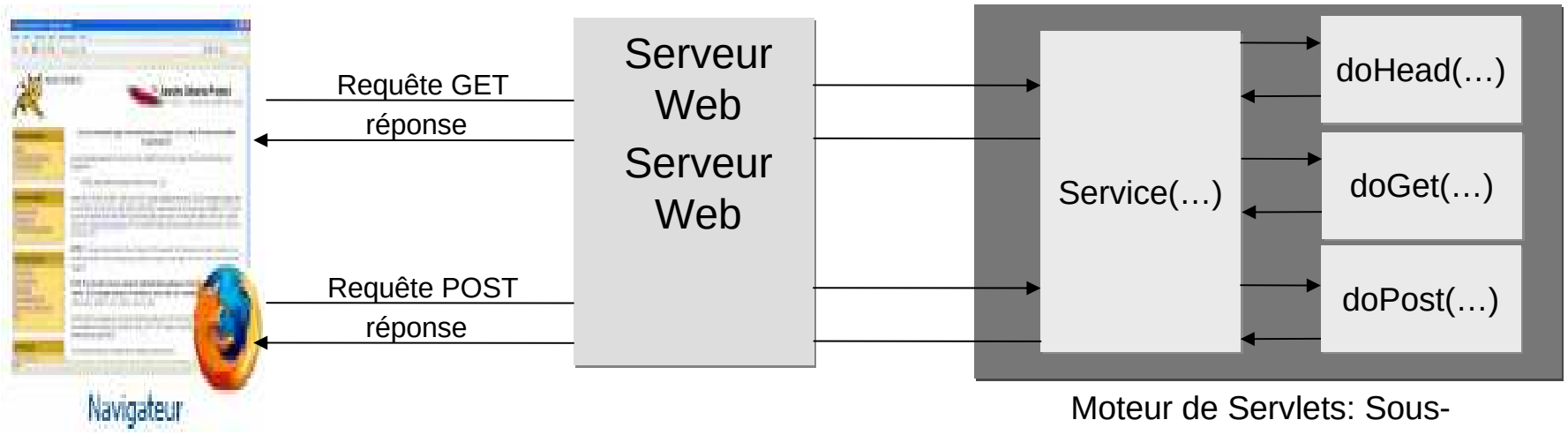


Servlet : exemple1



Servlet: HttpServlet

- Redéfinit la méthode service(…) qui:
  - lit la méthode (GET, POST…) à partir de la requête
  - transmet la requête a une méthode appropriée de HttpServlet (qui traite ce type de requête)



Servlet : exemple2

```
import java.io.*;
 import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
        public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
                res.setContentType("text/html");
                PrintWriter out = res.getWriter();
                out.println("<HTML>");
                out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
                out.println("<BODY>");
                out.println("<BIG>Hello World</BIG>");
                out.println("</BODY></HTML>");
        }
}
```

Hérite de HttpServlet

Http[ServletRequest, ServletResponse]

On implémente la méthode doGet

Servlet : HttpServletResponse

Utilisé pour construire un message de réponse HTTP renvoyé au client

Contient :

- les méthodes nécessaires pour définir le type de contenu, en-tête et code de retour
- un flot de sortie pour envoyer des données (par exemple HTML) au client

  - void setStatus(int) : définit le code de retour de la réponse
  - void setContentType(String) : définit le type de contenu MIME
  - PrintWriter getWriter() : permet d'envoyer des données texte au client
  - ServletOutputStream getOutputStream() : flot pour envoyer des données binaires au client
  - void sendRedirect(String) : redirige le navigateur vers l'URL

Servlet : HttpServletResponse

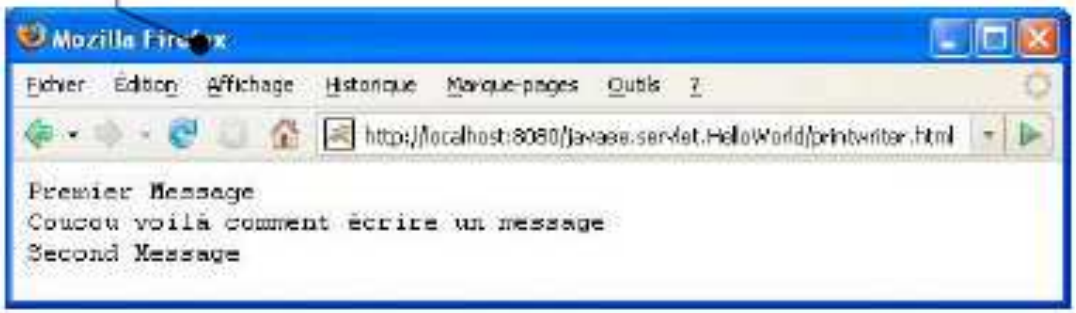
```
public class HelloWorldPrintWriter extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/plain");

        PrintWriter out = res.getWriter();

        out.println("Premier Message");
        out.println("Coucou voilà comment écrire un message");
        out.println("Second Message");
    }
}
```

HelloWorldPrintWriter.java



Servlet : HttpServletRequest

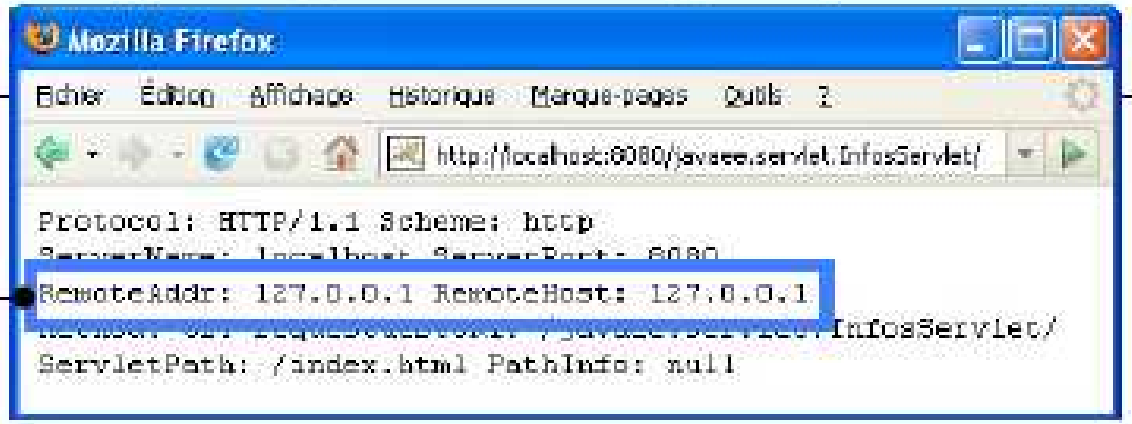
Encapsule la requête HTTP et fournit des méthodes pour :

- récupérer les paramètres passés au serveur par le client
- accéder aux informations du client et de l'environnement du serveur

  - String getMethod() : retourne le type de requête
  - String getServeurName() : retourne le nom du serveur
  - String getParameter(String name) : retourne la valeur d'un paramètre
  - String[] getParameterNames() : retourne le nom des paramètres
  - String getRemoteAddr() : retourne l'IP du client
  - String getServerPort() : retourne le port sur lequel le serveur écoute
  - …(voir l'API Servlets)

Servlet : HttpServletRequest

```
public class InfosServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        response.setContentType("text/plain");
        PrintWriter out= response.getWriter();
        out.println("Protocol: " + request.getProtocol());
        out.println("Scheme: " + request.getScheme());
        out.println("ServerName: " + request.getServerName());
        out.println("ServerPort: " + request.getServerPort());
        out.println("RemoteAddr: " + request.getRemoteAddr());
        out.println("Method: " + request.getMethod());
}}
```



InfosServlet.java

Servlet : Exercices

1. Quelles possibilités s'offrent à nous, si on ne veut pas différencier le type de la méthode utilisée pour la requête?

utiliser la classe GenericServlet ou implémenter uniquement la méthode service()

2. Proposez une implémentation des méthodes doGet et doPost si on veut avoir le même comportement pour ces deux types d'appel

   L'une appelle l'autre

3. Sans implémenter une méthode doHead, proposez une implémentation pour supporter cette méthode.

Servlet : Exercices

3. Sans implémenter une méthode doHead, proposez une implémentation pour supporter cette méthode.

en détectant les requêtes de type HEAD dans la méthode doGet():

```
public void doGet(HttpServletRequest req, HttpServletResponse res)throws ServletException, IOException {

  // l'attribut Content-Type  de l'entête (Head)
  res.setContentType("text/html");

  // Si la méthode est HEAD renvoyer le résultat(ici Content-Type  de l'entête)   if (req.getMethod().equals("HEAD")) return;

  // Sinon, corps de la servlet
  PrintWriter out = res.getWriter();
  out.println("<HTML>");
  .....}
```

Servlet : Exercices

Ecrire une servlet qui selon la valeur du paramètre « action » reçu:

=« erreur » :Affiche une page d'erreur

=« redirect »:Redirige vers « redirection.html »

=« page »: Affiche une page avec un texte de votre choix

HttpServletResponse

void setStatus(int) : définit le code de retour de la réponse
void setContentType(String) : définit le type de contenu MIME
PrintWriter getWriter() : permet d'envoyer des données texte au client
ServletOutputStream getOutputStream() : flot pour envoyer des données binaires au client
void sendRedirect(String) : redirige le navigateur vers l'URL

HttpServletRequest

String getMethod() : retourne le type de requête
String getServeurName() : retourne le nom du serveur
String getParameter(String name) : retourne la valeur d'un paramètre
String[] getParameterNames() : retourne le nom des paramètres
String getRemoteAddr() : retourne l'IP du client
String getServerPort() : retourne le port sur lequel le serveur écoute

Servlet et formulaires

- Utilisation de la balise <FORM> </FORM>
  - Option METHOD : type de requête GET ou POST
  - Option ACTION : l'URL où envoyer les données

- Utilisation de composants IHM pour saisir des informations
  - Contenu à l'intérieur de la balise FORM
  - Chaque composant est défini au moyen d'une balise particulière
    SELECT, INPUT, TEXTAREA, …

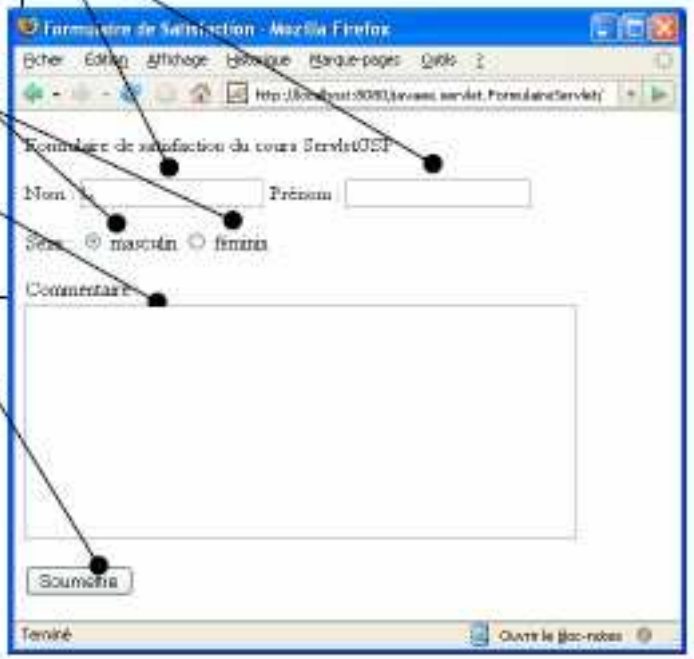
- Chaque composant a un nom (NAME) qui permet de l'identifier
    NAME="mon_composant "

- Les données sont envoyées quand l'utilisateur clique sur un bouton de type SUBMIT

Servlet et formulaires

```
<body>
<p>Formulaire de satisfaction du cours Servlet/JSP </p>
<form name="form1" method="get"
     action="/javaee.servlet.FormulaireServlet/form ">
  <p>
     Nom : <input type="text" name="nom">
     Prénom : <input type="text" name="prenom">
  </p>
  <p>
     Sexe :
     <input type="radio" name="radio1" value="sexe" checked>masculin
     <input type="radio" name="radio1" value="sexe">féminin
  </p>
  <p>Commentaire :<br>
     <textarea name="textarea" cols="50" rows="10"> </textarea><br>
     <input type="submit" name="Submit" rows="5" value="Soumettre">
  </p>
</form>
```

Le formulaire appelle une Servlet avec une requête de type GET



index.html

Servlet et formulaires

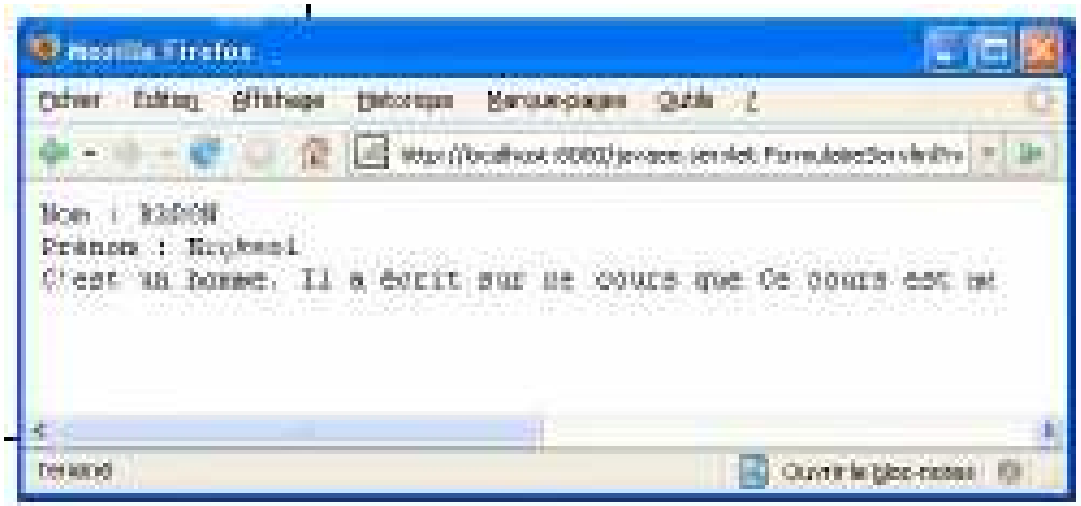
Accéder aux paramètres par l'intermédiaire de l'objet HttpServletRequest :

- String getParameter (String p) : retourne la valeur du paramètre p
- String[] getParamterValues (String p) : retourne les valeurs du paramètre p

Servlet et formulaires

Exercice:

Ecrire la servlet FormulaireServlet qui permet à partir d'une action sur le formulaire précédent d'affiche une page telle que:



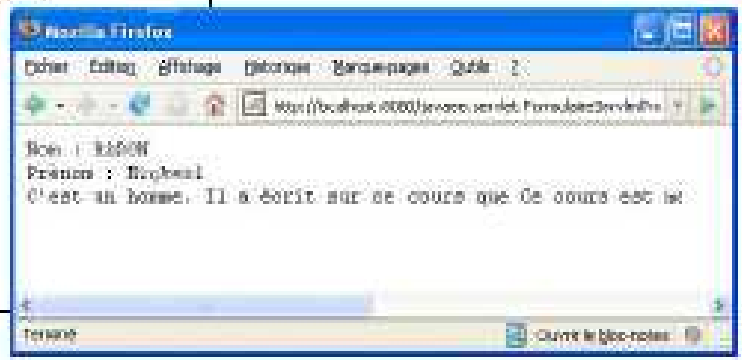
Servlet et formulaires

```
public class FormulaireServlet extends HttpServlet {
     protected void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
          res.setContentType("text/plain");
          PrintWriter out = res.getWriter();
          out.println("Nom : " + req.getParameter("nom"));
          out.println("Prénom : " + req.getParameter("prenom"));

          if (req.getParameterValues("radio1")[0].equals("mas")) {
            out.print("C'est un homme. Il");
          } else {
            out.print("C'est une femme. Elle");
          }

          out.print(" a écrit sur ce cours que ");
          out.println(req.getParameter("textarea"));
     }
}
```

FormulaireServlet.java



Servlet: suivi de session

- HTTP : protocole sans état

⇨ Impossibilité de garder des informations d'une requête à l'autre (identifier un client d'un autre)

⇨ Utilisation de différentes solutions pour remédier au problème d'état dont:

  - Cookies
  - HttpSession

Servlet : cookies

Rappel:

- cookie : information envoyée au navigateur (client) par un serveur WEB qui peut ensuite être relue par le client

- Lorsqu'un client reçoit un cookie, il le sauve et le renvoie ensuite au serveur chaque fois qu'il accède à une page sur ce serveur

Servlet : cookies

- Classe javax.servlet.http.Cookie pour utiliser les Cookies
  - Cookie(String name, String value) : construit un cookie
  - String getName() : retourne le nom du cookie
  - String getValue() : retourne la valeur du cookie
  - setValue(String new_value) : donne une nouvelle valeur au cookie
  - setMaxAge(int expiry) : spécifie l'âge maximum du cookie

Servlet : cookies

- Création d'un nouveau cookie
  - ⇨Ajout à la réponse (HttpServletResponse)
    - addCookie(Cookie mon_cook) : ajoute à la réponse un cookie
- Récupération des cookies du client
  - ⇨Récupération dans la requête (HttpServletRequest)
    - Cookie[] getCookies() : récupère l'ensemble des cookies du site

Servlet : cookies

- Code pour créer un cookie et l'ajouter au client

```
Cookie cookie = new Cookie("Id", "123");
res.addCookie(cookie);
```

- Code pour récupérer les cookies

```
Cookie[] cookies = req.getCookies();
if (cookies != null) {
    for (int i = 0; i < cookies.length; i++) {
        String name = cookies[i].getName();
        String value = cookies[i].getValue();
    }
}
```

Servlet : cookies

Exercice:

Implémenter la méthode doGet pour gérer les cookies:

- Si le client est déjà connu affichage d'un message: « Encore vous »
- Sinon attribution d'un cookie (sessionId) au client et affichage d'un message: « Bonjour le nouveau »

Utiliser java.rmi.server.UID().toString pour générer un identifiant

Servlet : cookies

```
public class CookiesServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
          throws ServletException, IOException {
        ...
        String sessionId = null;
        Cookie[] cookies = req.getCookies();
        if (cookies != null) {
            for (int i = 0; i < cookies.length; i++) {
                if (cookies[i].getName().equals("sessionid")) {
                    sessionId = cookies[i].getValue();
        }}}
        if (sessionId == null) {
            sessionId = new java.rmi.server.UID().toString();
            Cookie c = new Cookie("sessionid", sessionId);
            res.addCookie(c);
            out.println("Bonjour le nouveau");
        } else {
            out.println("Encore vous"); ... }
    }
}
```

Génère un identifiant unique pour chaque client

Servlet: HttpSession

API de suivi de session HttpSession

- Méthodes de création liées à la requête (HttpServletRequest)
  - HttpSession getSession() : retourne la session associée à l'utilisateur
  - HttpSession getSession(boolean p) : création selon la valeur de p

- Gestion d'association (HttpSession)
  - Enumeration getAttributNames() : retourne les noms de tous les attributs
  - Object getAttribut(String name) : retourne l'objet associé au nom
  - setAttribut(String na, Object va) : modifie na par la valeur va
  - removeAttribut(String na) : supprime l'attribut associé à na

- Destruction (HttpSession)
  - invalidate() : expire la session
  - logout() : termine la session

Servlet: HttpSession

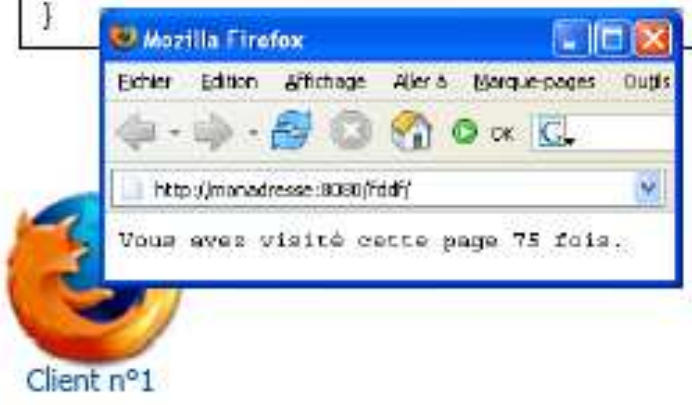
Exercice:

Implémenter la méthode doGet pour gérer le suivi de session:

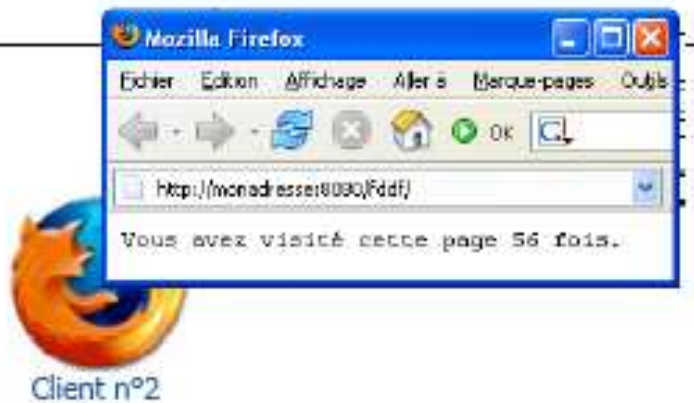
Affiche un compteur qui est incrémenté à chaque accès sur cette servlet pendant une session.

Servlet: HttpSession

```
public class HttpSessionServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
          throws ServletException, IOException {
        res.setContentType("text/plain"); PrintWriter out = res.getWriter();
        HttpSession session = req.getSession();
        Integer count = (Integer)session.getAttribute("count");
        if (count == null)
          count = new Integer(1);
        else
          count = new Integer(count.intValue() + 1);
        session.setAttribute("count", count);
        out.println("Vous avez visité cette page " + count + " fois.");
    }
}
```

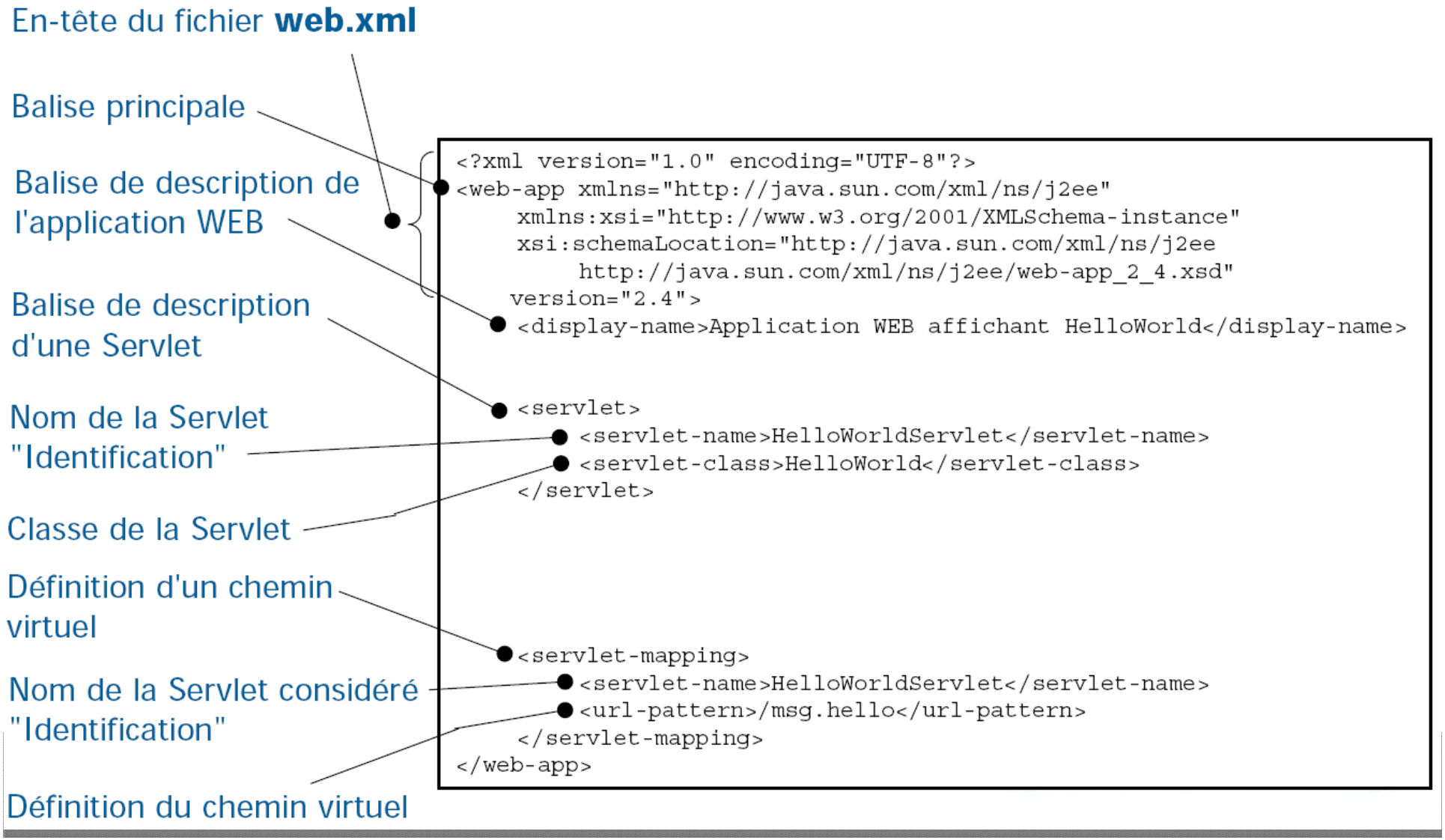


Client n°1

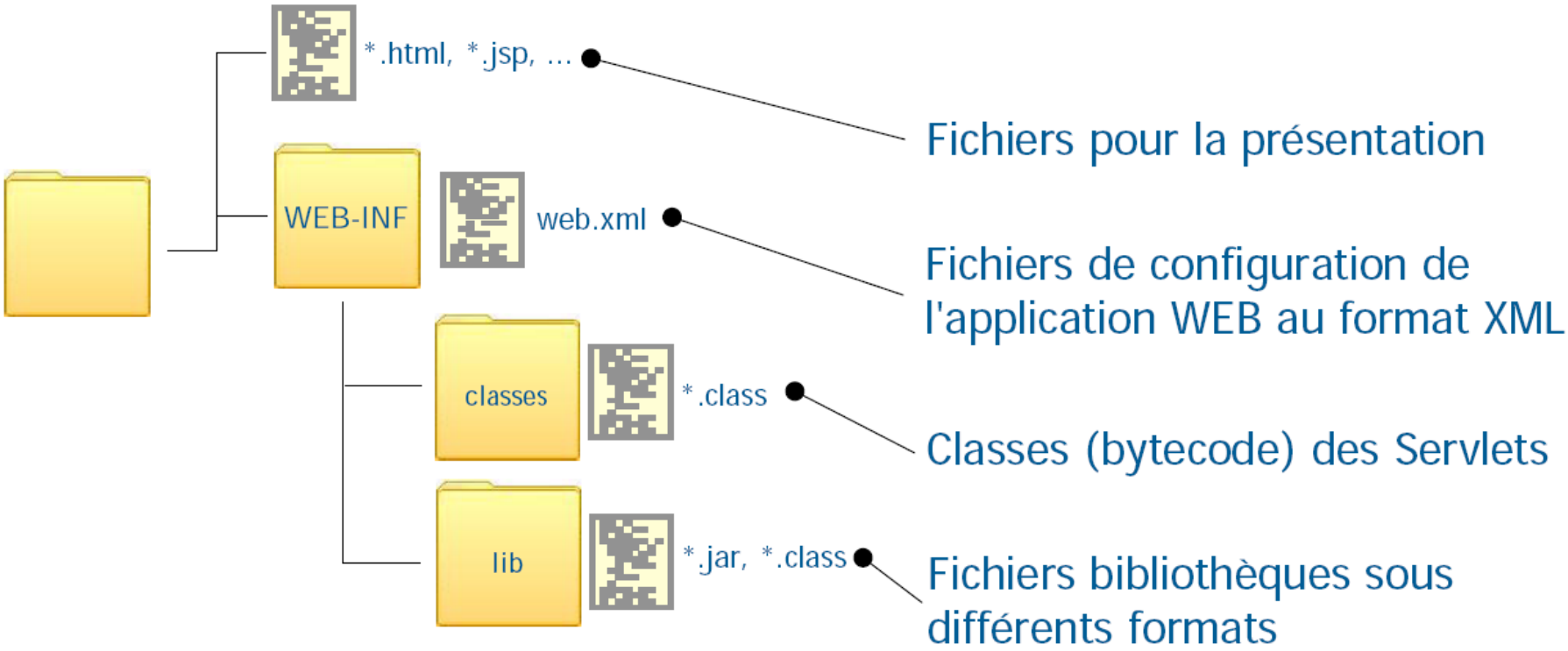


Client n°2

Servlet: déploiement



Servlet: déploiement



A venir : Les JSP