جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم والتقنيات
FACULTÉ DES SCIENCES ET TECHNIQUES

Module : JEE

Servlet

Chapitre III: Servlet(suite)

I-RequestDispatcher

- Redirection
- Inclusion

II-Suivit de Session

- Cookies
- Champ caché
- Réécriture de lien
- HttpSession

Chapitre III: Servlet(suite)

I. RequestDispatcher

L'interface **RequestDispatcher** permet de dispatcher la requête vers une autre ressource, qu'elle soit HTML, Servlet ou JSP. Cette interface peut également être utilisée pour inclure le contenu d'une autre ressource. C'est l'un des moyens de collaboration entre servlets.

Chapitre III: Servlet(suite)

I. RequestDispatcher

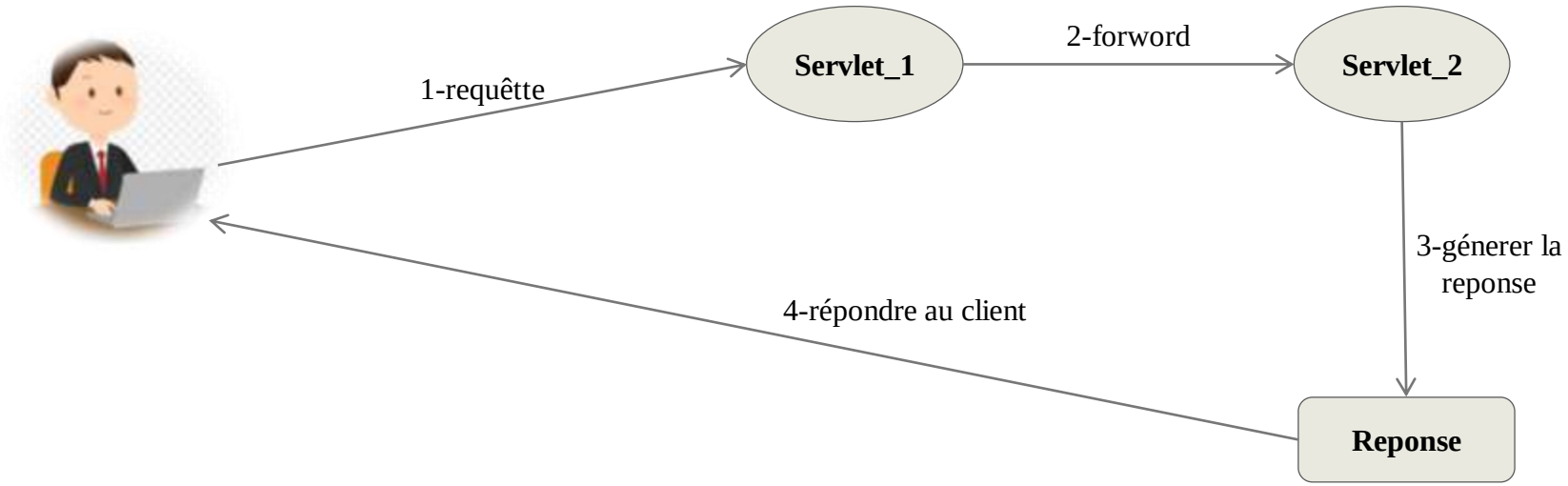
L'interface RequestDispatcher a deux méthodes:

- public void forward(ServletRequest rqt,ServletResponse rep) : Transmet une requête d'une servlet à une autre ressource (servlet, fichier JSP ou fichier HTML) sur le serveur.
- public void include(ServletRequest rqt,ServletResponse rep) : Inclut le contenu d'une ressource (servlet, page JSP ou fichier HTML) dans la réponse.

Chapitre III: Servlet(suite)

I. RequestDispatcher

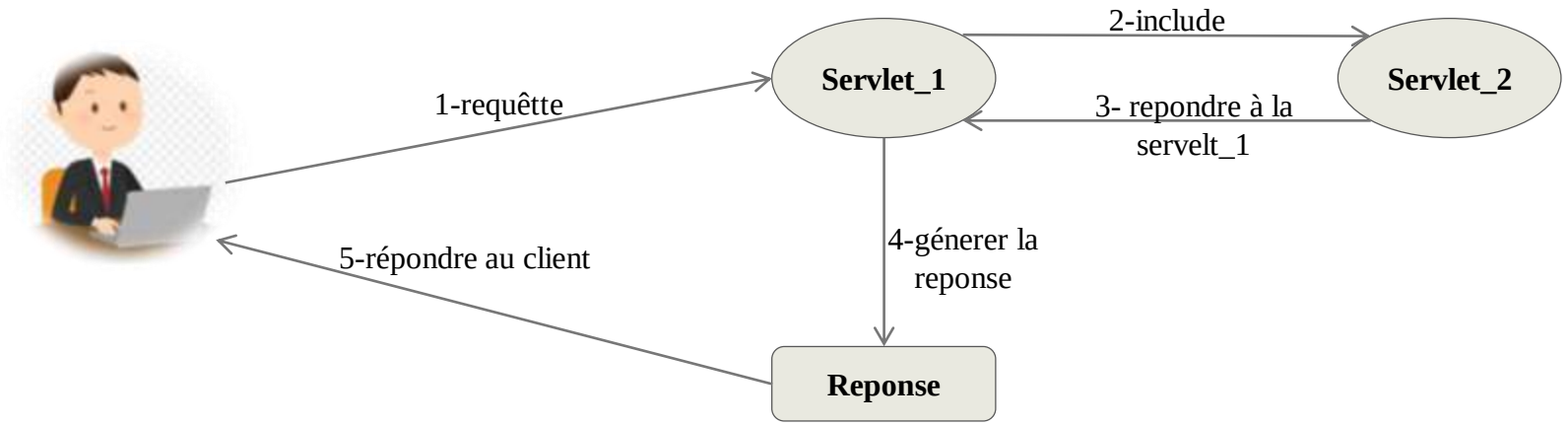
public void **forward**(ServletRequest rqt,ServletResponse rep)



Chapitre III: Servlet(suite)

I. RequestDispatcher

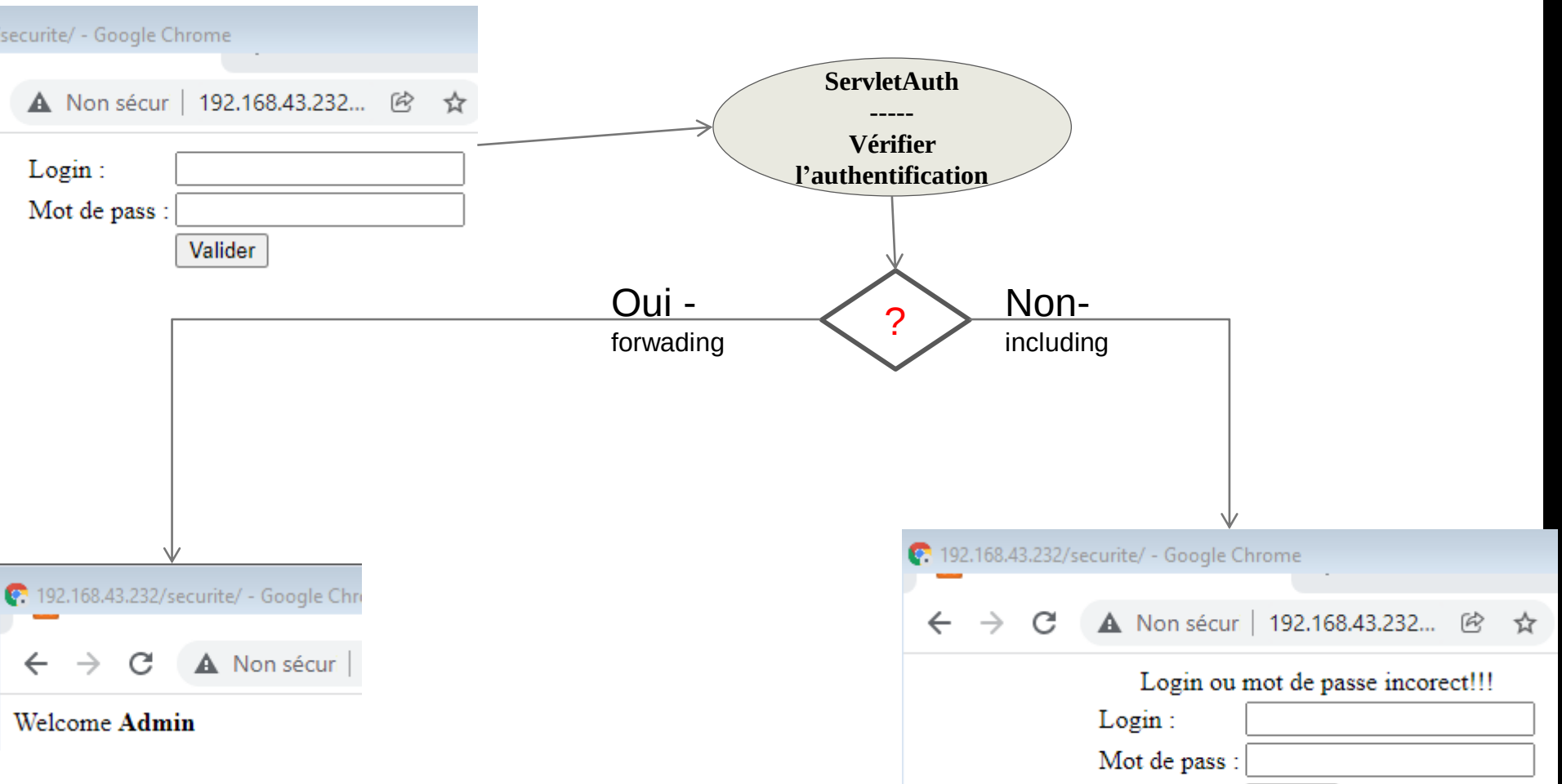
public void include(ServletRequest rqt,ServletResponse rep)



Chapitre III: Servlet(suite)

I. RequestDispatcher

Exemple : Authentificaion



Chapitre III: Servlet(suite)

I. RequestDispatcher

Exemple : Authentificaion

```
public class servletAuth extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    String l=request.getParameter("login");
    String p=request.getParameter("pass");

    if(l.equals("admin") && p.equals("admin123")){
        RequestDispatcher rd=request.getRequestDispatcher("servlet2");
        rd.forward(request, response);
    }
    else{
        out.print("Login ou mot de passe incorect!!!");
        RequestDispatcher rd=request.getRequestDispatcher("/index.html");
        rd.include(request, response);
        }
    }
```

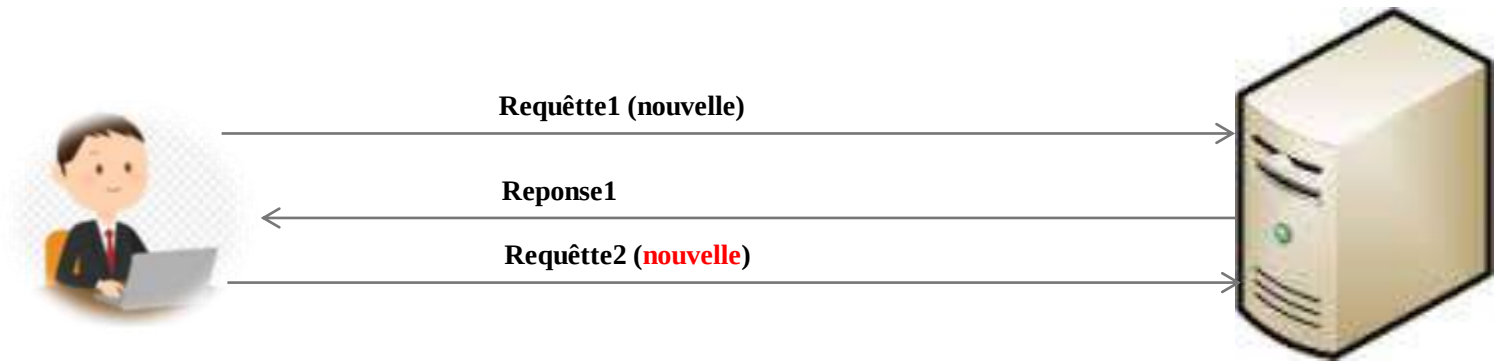
forward() vs sendRedirect()

Il existe de nombreuses différences entre la méthode **forward**() de **RequestDispatcher** et la méthode **sendRedirect**() de l'interface **HttpServletResponse**.

forward()	sendRedirect()
The forward() method works at server side.	The sendRedirect() method works at client side.
It sends the same request and response objects to another servlet.	It always sends a new request.
It can work within the server only.	It can be used within and outside the server.

II. Les sessions dans les servlets

Le protocole **HTTP** est un protocole sans état (stateless). Chaque fois qu'un utilisateur envoie une requête au serveur, ce dernier la traite comme une nouvelle requête.



Comment garder la trace d'un utilisateur pour le reconnaître ?

II. Les sessions dans les servlets

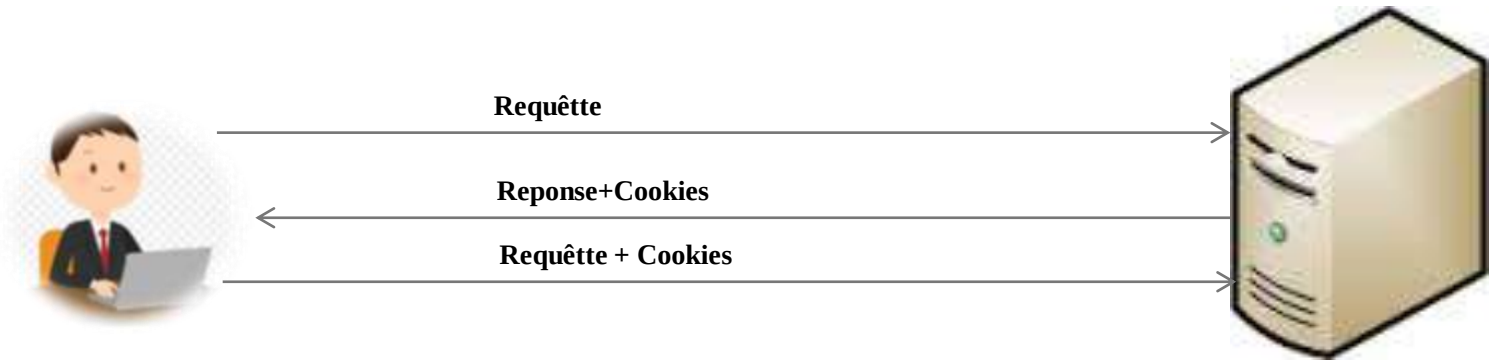
Il existe quatre techniques utilisées pour le suivi des traces des utilisateurs :

- Les cookies
- Champ de formulaire caché
- Réécriture d'URL
- HttpSession

II. Les sessions dans les servlets

1- Cookies

Un cookie est un petit élément d'information qui est conservé entre les multiples requêtes du client.



II. Les sessions dans les servlets

1- Cookies

Un cookie a un **nom**, une **valeur unique** et des attributs facultatifs tels qu'un ***commentaire***, des ***qualificatifs de chemin*** et de ***domaine***, un ***âge maximal*** et un ***numéro de version***.

Alors, la classe **javax.servlet.http.Cookie** a des méthodes suivantes:

getName() getValue() getComment() getPath() getDomaine() getMaxAge() getVersion()

setName() setValue() setComment() setPath() setDomaine() setMaxAge() setVersion()

II. Les sessions dans les servlets

1- Cookies

Pour ajouter un cookie ou obtenir la valeur d'un cookie, nous avons besoin de deux méthodes fournies par d'autres interfaces.

public void addCookie(Cookie ck):méthode de l'interface HttpServletResponse utilisée pour ajouter un cookie dans l'objet de réponse.

public Cookie[] getCookies(): méthode de l'interface HttpServletRequest utilisée pour retourner tous les cookies du navigateur.

II. Les sessions dans les servlets

1- Cookies

Ajouter les cookies dans une réponse

```
//creation de deux cookies
Cookie ck_user=new Cookie("user","azrour");//creer cookie
Cookie ck_pw=new Cookie("pass","Passe123");

//ajout des cookies dans la reponse
response.addCookie(ck_user);
response.addCookie(ck_pw);
```

II. Les sessions dans les servlets

1- Cookies

Récupérer les cookies d'une requêtte

```
//recuper les cookies de la requête
  Cookie listcook[]=request.getCookies();
    for(int i=0;i<listcook.length;i++){
        //afficher nom et valeur des cookies
        out.print("<br>"+listcook[i].getName()+" "+listcook[i].getValue());
    }
```

II. Les sessions dans les servlets

2- Champ de formulaire caché

Dans ce cas un champ de formulaire caché (invisible) est utilisé pour maintenir l'état (paramètres…) d'un utilisateur.

```
//recuprer le parametre 'userName'
String nom=request.getParameter("userName");
out.print("Bonjour "+nom);

//creer un formulaire avec champ caché
out.print("<form action='servlet2'>");
out.print("<input type='hidden' name='uname' value='"+nom+"'>");
out.print("<input type='submit' value='go'>");
out.print("</form>");
```

II. Les sessions dans les servlets

3- Réécriture d'URL

Dans la réécriture d'URL, nous ajoutons un jeton ou un identifiant à l'URL de la prochaine servlet ou de la prochaine ressource. Nous pouvons envoyer des paires nom/valeur de paramètres en utilisant le format suivant :

url?name1=value1&name2=val

```
//recuprer le parametre 'userName'
String nom=request.getParameter("username");
out.print("Bonjour "+nom);

//réécrir une URL en lui associant un parametre 'username'
out.print("<a href='servlet2?uname="+nom+"'>visit</a>");
```

II. Les sessions dans les servlets

4- L'interface HttpSession

Dans ce cas, le conteneur crée un identifiant de session pour chaque utilisateur. Le conteneur utilise cet identifiant pour identifier l'utilisateur particulier. Un objet de **HttpSession** peut être utilisé pour effectuer deux tâches :

- Lier les objets
- Afficher et manipuler les informations relatives à une session, telles que **l'identifiant** de la session, **l'heure** de création et **l'heure** du dernier accès.

II. Les sessions dans les servlets

4- L'interface HttpSession

Comment obtenir l'objet HttpSession ?

L'interface HttpServletRequest fournit deux méthodes pour obtenir l'objet de HttpSession :

public HttpSession getSession(): Retourne la session actuelle associée à cette requête, ou si la requête n'a pas de session, en crée une .

public HttpSession getSession(boolean valeur): Renvoie la session HttpSession actuelle associée à cette requête ou, s'il n'y a pas de session actuelle et que le paramètre est **true**, renvoie une nouvelle session.

II. Les sessions dans les servlets

4- L'interface HttpSession

Méthodes fréquemment utilisées de l'interface HttpSession

public String getId():Retourne une chaîne contenant la valeur de l'identifiant unique.

public long getCreationTime():Retourne l'heure à laquelle cette session a été créée, mesurée en millisecondes depuis minuit le 1er janvier 1970 GMT.

public long getLastAccessedTime():Renvoie l'heure à laquelle le client a envoyé pour la dernière fois une requête associée à cette session, en nombre de millisecondes depuis minuit le 1er janvier 1970 GMT.

public void invalidate():Invalide cette session puis libère les objets qui lui sont liés.

II. Les sessions dans les servlets

4- L'interface HttpSession

```
public class servletAuth extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

        //recuprer le parametre 'userName'
        String nom=request.getParameter("username");
        String pss=request.getParameter("passwd");

        if (nom.equals("admin") && pss.equals("123") ){
            HttpSession session=request.getSession();
            session.setAttribute("authentifier","oui");
            out.println("Authetification avec succes");

             //lien vers servlet2
            out.print("<a href=/site3/contact> page servlet2 </a>");

        }else{
            out.println("Authetification echoué <br>  réssayer une autre fois");
            request.getRequestDispatcher("enregistrement.html").include(request, response);
                //redirection vers la page d'authentification
        }
    }
```

II. Les sessions dans les servlets

4- L'interface HttpSession

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      PrintWriter out = response.getWriter();

        HttpSession session=request.getSession(false);
        String auth=(String)session.getAttribute("authentifier");
        if (auth.equals("oui")){
             out.print("Vous êtes authentifié ");

             //afficher le contenu de cette page
        }else{
             out.print("<b><font color='#990011'> Cette page est réserver au personne  authentifié</font></b>");
                //redirection vers la page d'authentification
        }
   }
```

Tp_3_Authentification et session