

Université Constantine 2 Abdelhamid Mehri

Faculté des Nouvelles Technologies

Département Technologies des Logiciels et Systèmes d'Information

LES SERVLETS

DR. zakaria LAKHDARA / zakaria.lakhdara@univ-constantine2.dz



Faculté
NTIC

Département
TLSI

Niveau
L3

Spécialité
GL

SERVLET

- Une **servlet** représente la **partie contrôle** dans une architecture MVC.
- Une servlet est **un programme JAVA qui s'exécute côté serveur**.
- Une servlet représente une extension des fonctionnalités du serveur web.
- Elle peut être invoquée par un navigateur via une URL à travers le protocole de communication HTTP.
- Une servlet traite les requête HTTP des client et génère des pages html dynamiques comme réponse.

RÔLE DES SERVLETS

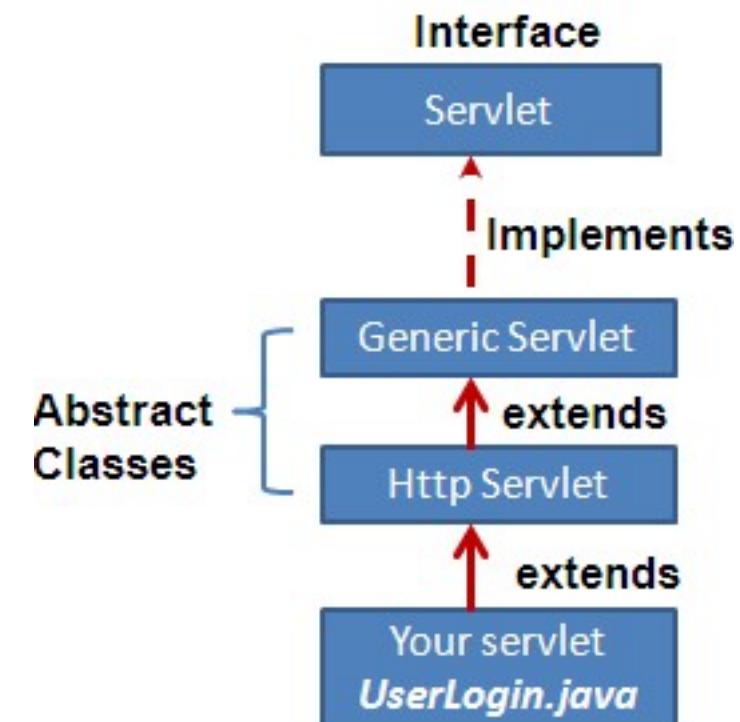
- Recevoir la requête HTTP du client.
- Déclencher les traitements demandés et récupère les résultats.
- Identifier les pages HTML dynamiques pour renvoyer les résultats au client.

API SERVLETS

- l'API Servlet est un ensemble d'interfaces et de classes Java, rangées dans les packages

javax.servlet et **javax.servlet.http**.

- **javax.servlet** : contient les classes pour développer des servlets génériques indépendantes d'un protocole (protocole requête/réponse).
- **javax.servlet.http** : contient les classes pour développer des servlets qui reposent sur le protocole http utilisé par les serveurs web.



SERVLET

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

@WebServlet("/myServlet")
public class myServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public myServlet() {
        super();
    }

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

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    }
}
```

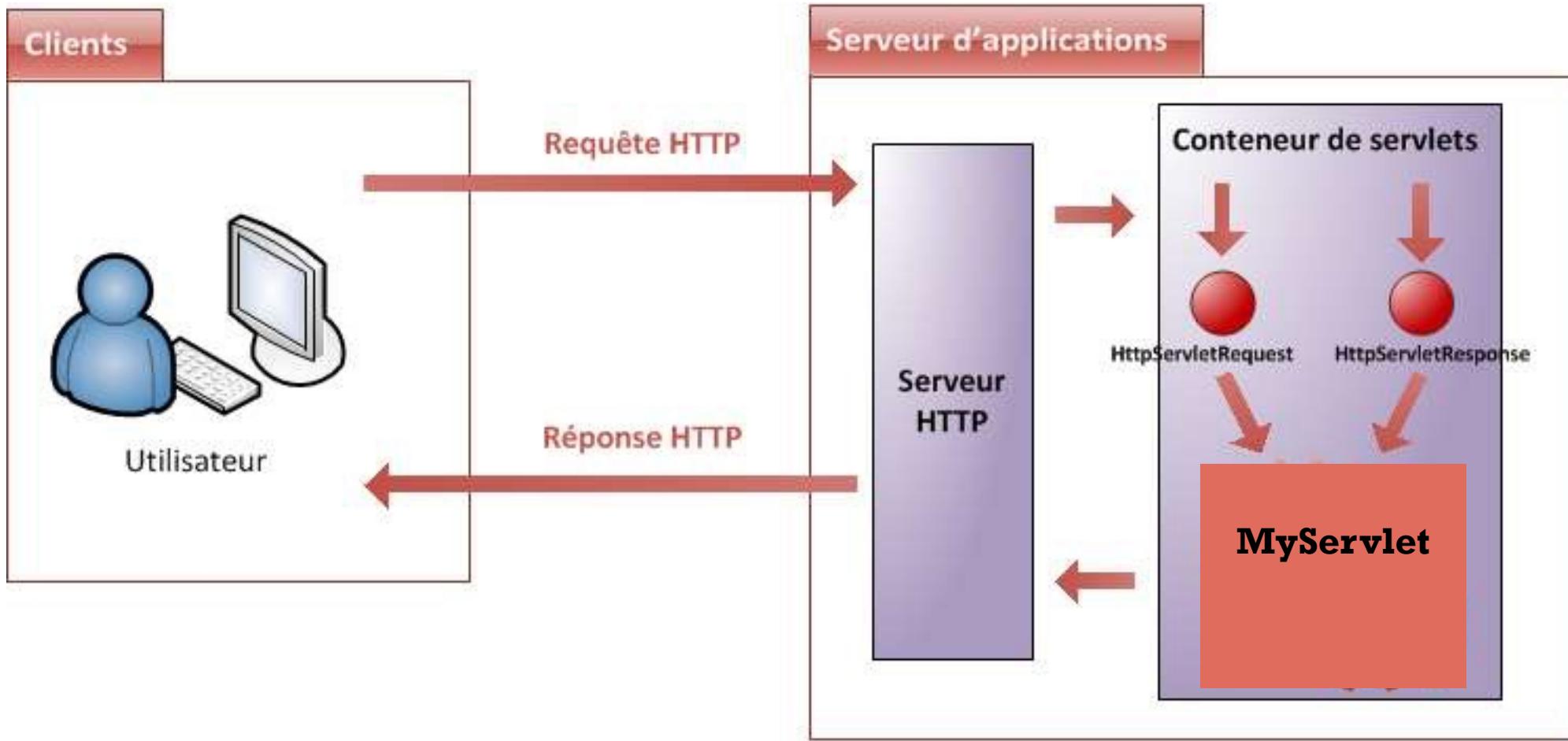
SERVLETS

- La servlet **MyServlet** définie est une sous-classe de **HttpServlet**
- Surcharger les méthodes **doGet()** ou **doPost()**, en fonction du mode d'envoi des données utilisés(par GET ou POST).
- Ces deux méthodes prennent deux arguments (deux objets) :
request HttpServletRequest et **response HttpServletResponse**.

SERVLETS

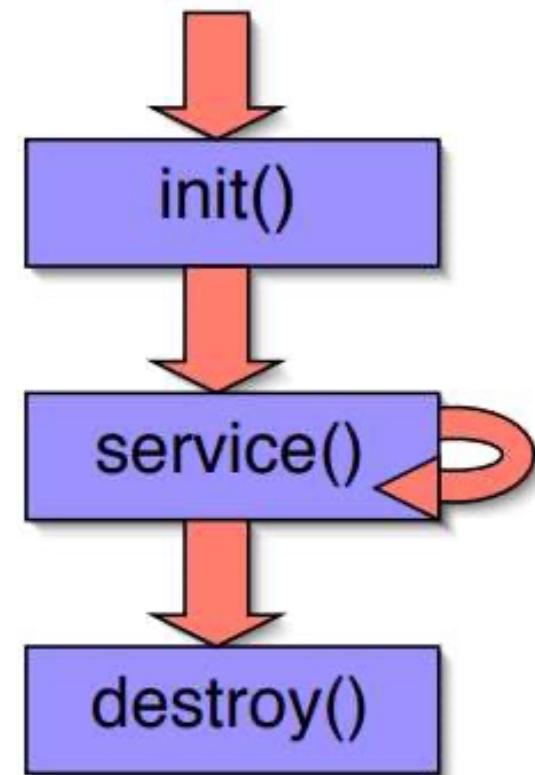
- **request** de type **HttpServletRequest**
 - Contient les informations de la requête (données d'un formulaire)
 - Offre des méthodes pour récupérer ces informations
- **response** de type **HttpServletResponse**.
 - Contient la réponse HTTP qui sera renvoyée au client, et permet de la personnaliser.

SERVLETS



CYCLE DE VIE D'UNE SERVLETS

- Première requête:
 - Chargement de la classe dans JVM.
 - Instanciation de la servlet
 - Exécution de **init()**
- Toute requête:
 - Crédation des objets **request** et **response**
 - Exécution de **service()**
- Fin de vie :
 - Exécution de **destroy()**



SERVLETS

- Le conteneur web géré le cycle de vie des servlets en exécutant les méthodes `init()`, `service()`, et `destroy()`.
- **init()** est appelée par le conteneur web juste après l'instanciation de la servlet.
- **service()** ne peut pas être invoquée tant que la méthode `init()` n'est pas terminée. Elle réalise le traitement demandé
- **destroy()** est appelée pour détruire la servlet. Cela permet de libérer les ressources allouées dans la méthode `init()`.

TRAITEMENT DES REQUÊTES HTTP



TRAITEMENT DES REQUÊTES HTTP

- GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE et CONNECT.
 - **doGet()** : pour les requêtes http de type **GET**
 - **doPost()** : pour les requêtes http de type **POST**
 - doHead() : pour les requêtes http de type HEAD
 - doPut() : pour les requêtes http de type PUT
 - doDelete() : pour les requêtes http de type DELETE
 - doOptions() : pour les requêtes http de type OPTIONS
 - doTrace() : pour les requêtes http de type TRACE

TRAITEMENT DES REQUÊTES HTTP

- La méthode **service()** est appelée lors de l'invocation de la servlet.
- Cette méthode réalise une analyse de la requête client contenue dans l'objet **HttpServletRequest request**.
- Selon le type de requête GET ou POST, elle appelle
 - **doGet()** : pour les requêtes http de type **GET**
 - **doPost()** : pour les requêtes http de type **POST**

TRAITEMENT DES REQUÊTES HTTP

LA MÉTHODE **DOGET()**

- Utilisée pour traiter les requêtes de type GET
- Pour la récupération des ressources

Utilisée par exemple:

- En cliquant sur un lien

```
<A HREF="http://localhost:8080/myWebApp/FormServlet "> Contact us form </A>
```

- En entrant l'URL brute dans la barre d'adresse du navigateur

`http://localhost:8080/myWebApp/FormServlet`

TRAITEMENT DES REQUÊTES HTTP

LA MÉTHODE **DOPOST()**

- Utilisée pour traiter les requêtes de type POST
- Pour la collecte de données à partir d'un formulaire HTML
- L'exécution de tâches (enregistrement dans une base de données)

Contact us

Your name *

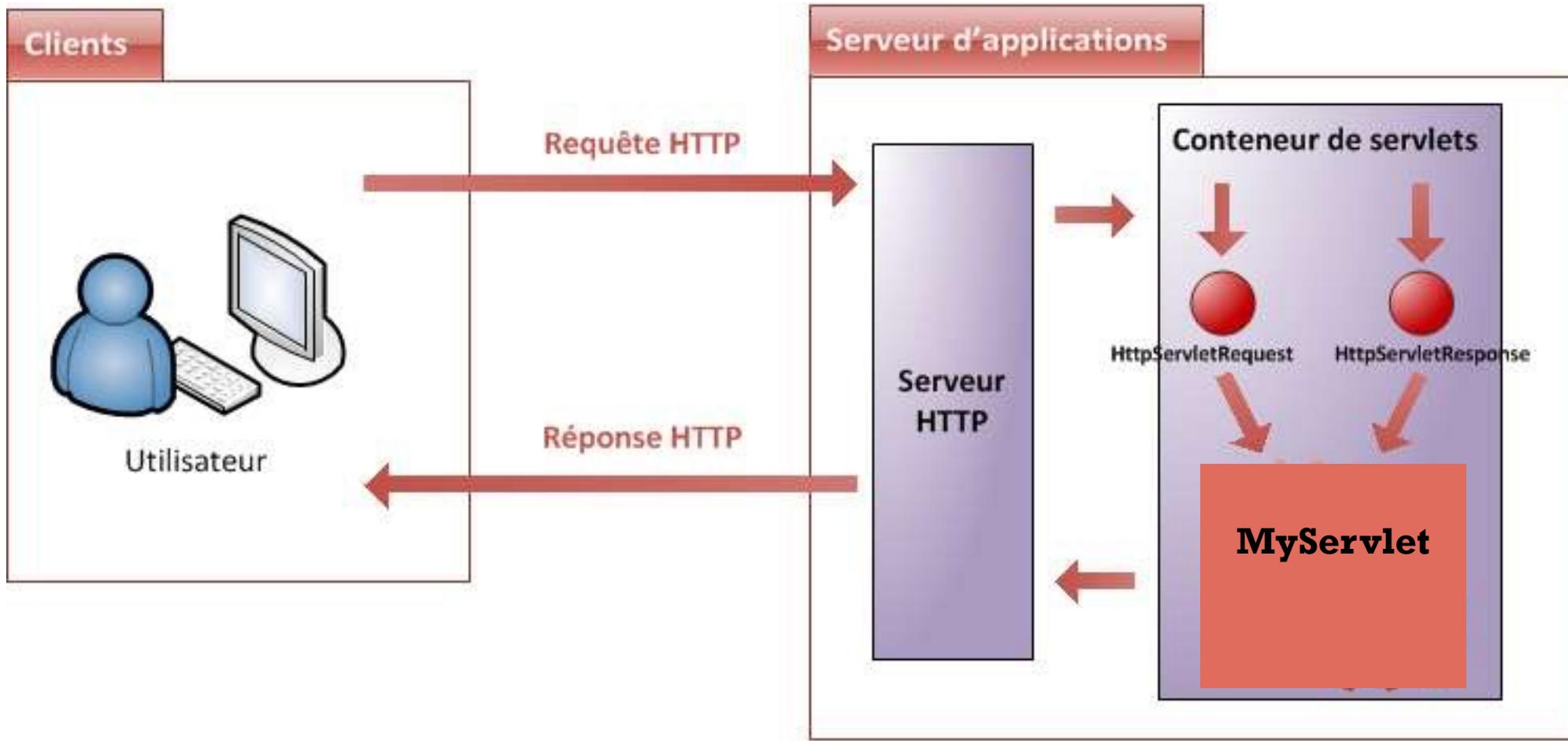
Your e-mail address *

Message *

Send

Reset

SERVLETS



TRAITEMENT DES REQUÊTES HTTP: REQUEST

- L'objet **HttpServletRequest request** encapsule la requête HTTP et fournit des méthodes pour accéder à toutes les informations nécessaires.
- Parmi les méthodes proposées par l'interface **ServletRequest** :
 - **getParameter()**: permet de récupérer la valeur d'un paramètre.
 - **setAttribute()** : permet de déposer un paramètre dans l'objet request
 - **getAttribute()** : permet de récupérer un paramètre de l'objet request

TRAITEMENT DES REQUÊTES HTTP: REQUEST

- **getParameter():**

```
public class Message {  
  
    private int id;  
    private String name;  
    private String email;  
    private String message;  
  
    public Message(HttpServletRequest request) {  
        this.id = 0;  
        this.name = request.getParameter("name");  
        this.email = request.getParameter("email");  
        this.message = request.getParameter("message");  
    }  
}
```

TRAITEMENT DES REQUÊTES HTTP: REQUEST

- **setAttribute()** : permet de déposer un paramètre dans l'objet request

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    //create a new message myMsg
    Message myMsg = new Message(request);

    try {

        //insert myMsg in the db and change its state to saved
        MessageDAO myMsgDAO = new MessageDAO();
        myMsgDAO.saveMessage(myMsg);

    } catch (InstantiationException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }

    // send myMsg to formResult.jsp

    request.setAttribute("myMsg", myMsg);

    this.getServletContext().getRequestDispatcher("/WEB-INF/formResult.jsp").forward(request, response);
}
```

TRAITEMENT DES REQUÊTES HTTP: **RESPONSE**

- L'objet **HttpServletResponse response** est utilisé pour construire un message de réponse HTTP renvoyé au navigateur client.
- Il contient les méthodes nécessaires pour définir le type de contenu **setContentType("text/html")**

Il contient aussi un flot de sortie pour envoyer des données (HTML ou autre) au navigateur.

PrintWriter

getWriter()

TRAITEMENT DES REQUÊTES HTTP: **RESPONSE**

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

    response.setContentType("text/html");

    PrintWriter out =response.getWriter();

    out.println("<html>");
    out.println("<body>");
    out.println("<head>");
    out.println("<title> Example of a response </title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<p> hello everybody !!! </p>");
    out.println("</body>");
    out.println("</html>");

}
```

TRAITEMENT DES REQUÊTES HTTP: REQUESTDISPATCHER

- L'objet **RequestDispatcher** est utilisé pour dispatcher la requête à une autre ressource (html, jsp ou servlet)
- Utiliser la méthode **getRequestDispatcher(String)**

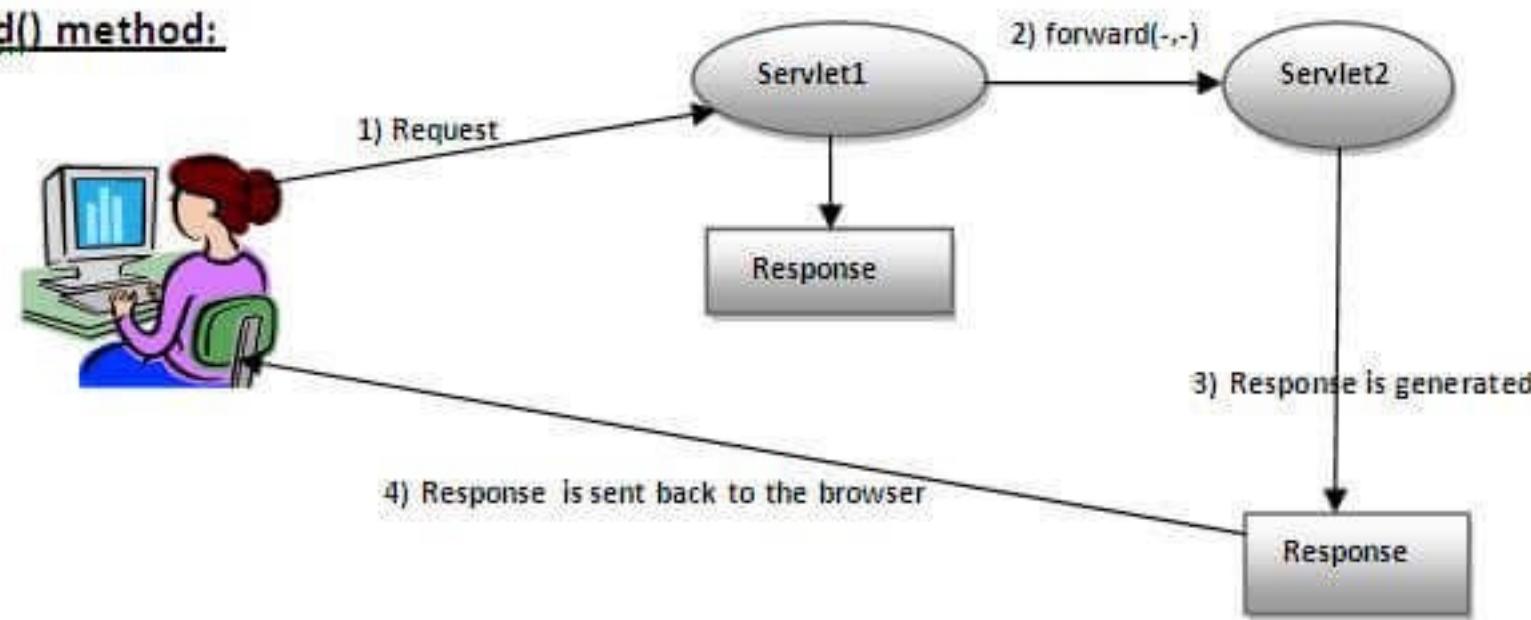
```
RequestDispatcher requestDispatcher =  
    request.getRequestDispatcher( "ressource" ) ;
```

- L'interface RequestDispatcher propose deux méthodes.
forward() et include()

TRAITEMENT DES REQUÊTES HTTP: REQUESTDISPATCHER

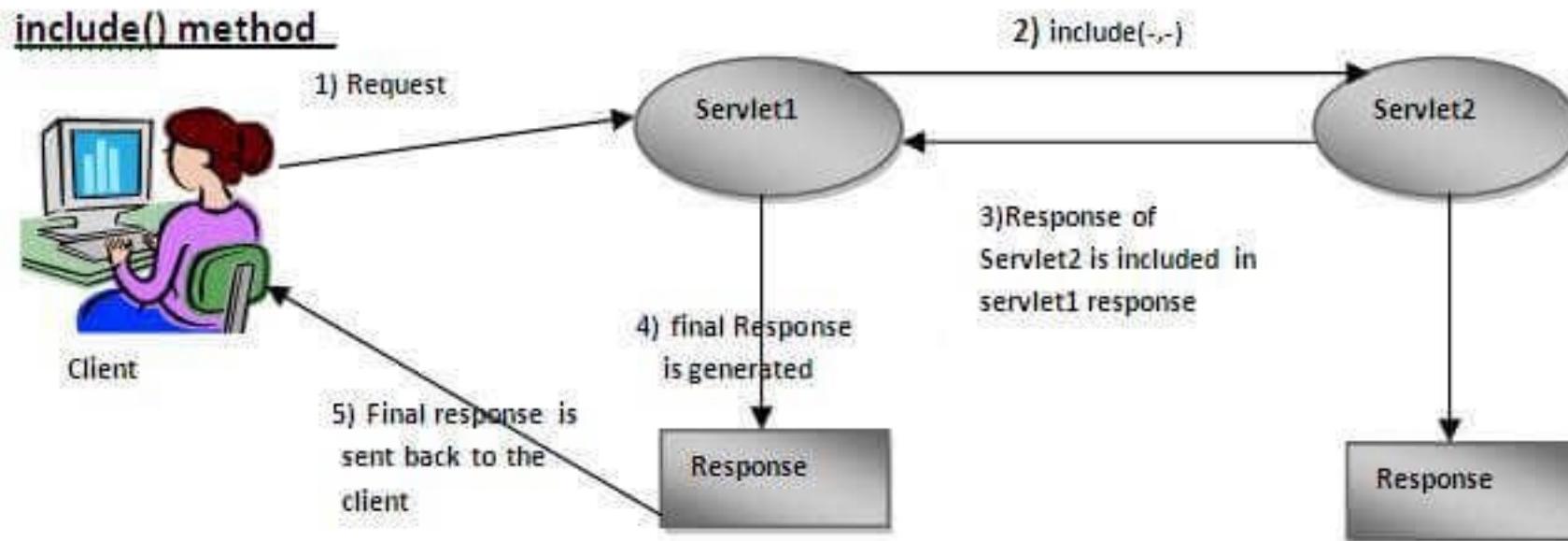
- **forward():** transfère une demande d'un servlet vers une autre ressource (servlet, fichier JSP ou fichier HTML) sur le serveur.

forward() method:

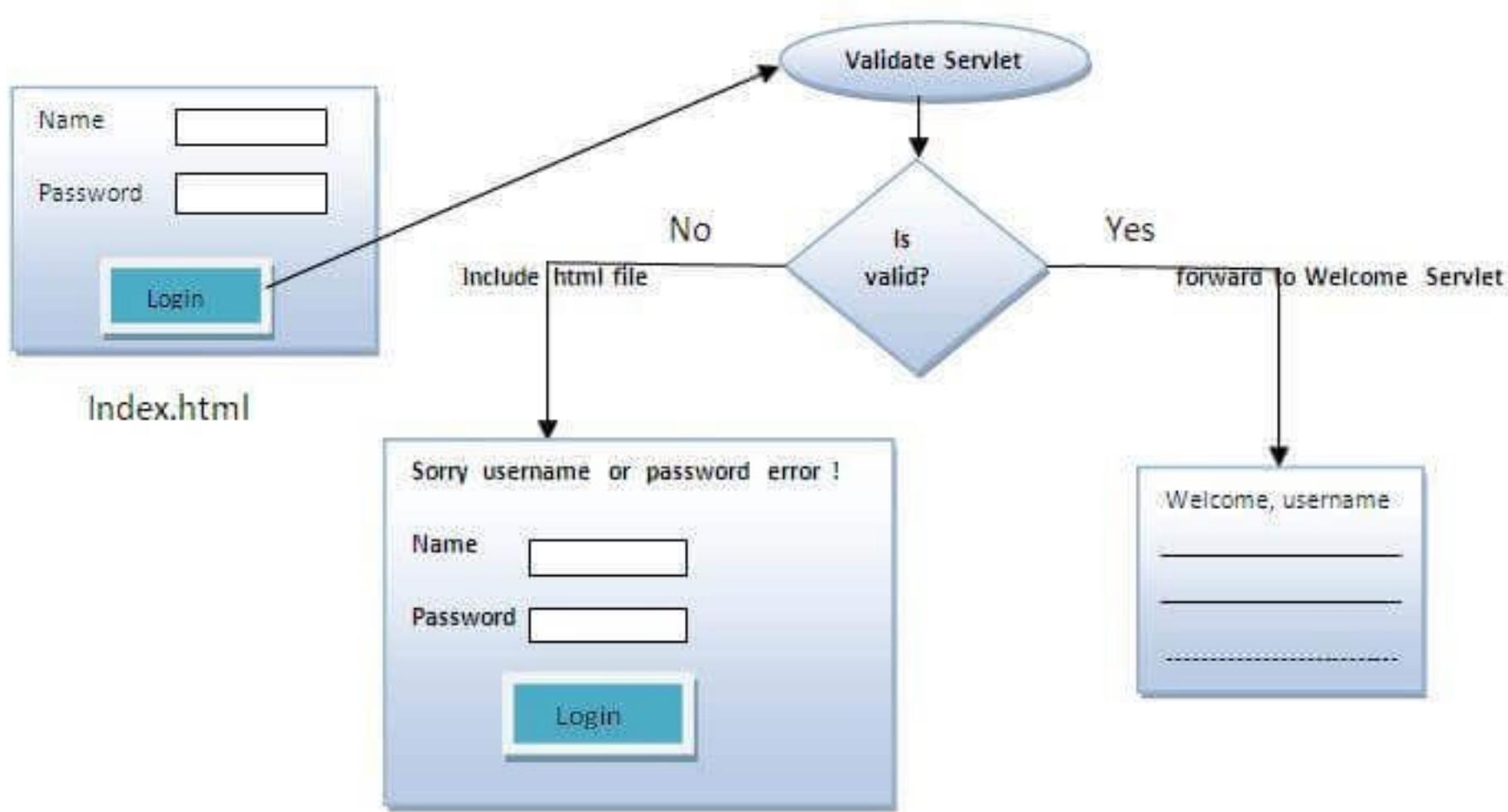


TRAITEMENT DES REQUÊTES HTTP: REQUESTDISPATCHER

- **include()**: inclut le contenu d'une ressource (servlet, page JSP ou fichier HTML) dans la réponse.



TRAITEMENT DES REQUÊTES HTTP: REQUESTDISPATCHER



CONCEPT DE SESSION



CONCEPT DE SESSION

- Le protocole HTTP est un **protocole non connecté** (on parle aussi de **protocole sans états**, en anglais stateless protocol).
- Le serveur ne conserve pas les données d'un client, une fois la réponse envoyée.
- Le serveur web ne peut pas se "souvenir" de la requête précédente.
- Le traitement des requêtes clientes est réalisé requête par requête le serveur web n'offre par défaut aucun suivi de session.

CONCEPT DE SESSION

Une SESSION:

- Représente un espace mémoire alloué pour chaque utilisateur, pour sauvegarder des informations .
- Permet au serveur de **mémoriser** les informations relatives au client.
- Le contenu d'une session est conservé jusqu'à ce que l'utilisateur ferme son navigateur, reste inactif trop longtemps, ou il se déconnecte.
- Cette fonctionnalité est supportée par les servlets à l'aide de l'objet

HttpSession

CONCEPT DE SESSION

- Créer un objet **session** en utilisant la méthode **getSession()** de l'objet **HttpServletRequest request**
HttpSession session = request.getSession()
- Utiliser l'objet **session** pour stocker les informations du client en utilisant la méthode
setAttribute(String name, Object value)
- Récupérer les informations du client en utilisant la méthode
getAttribute(String name)

CONCEPT DE SESSION

Servlet 1: Creation d'une session

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    String user = request.getParameter("userName");
    String password = request.getParameter("userPassword");

    HttpSession session = request.getSession();

    session.setAttribute("uname", user);
    session.setAttribute("upass", password);
}
```

CONCEPT DE SESSION

Servlet 2: récupération de la session dans une 2eme servlet

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter writer = response.getWriter();
    HttpSession session = request.getSession(false);
    String myName = (String) session.getAttribute("uname");
    String myPass = (String) session.getAttribute("upass");
    writer.println("Name: " + myName + " Pass: " + myPass);
    writer.close();
}
```