Les servlets

Cours 3

Présentation des servlets, L'API servlet, Le protocole HTTP, Les servlets HTTP, Les cookies

Université Internationale de Casablanca

Présentation du chapitre

- La présentation des servlets
- L'API servlet
- Le protocole HTTP
- Les servlets http
- Les informations sur l'environnement d'exécution des servlets
- L'utilisation des cookies
- Le partage d'informations entre plusieurs échanges HTTP
- Packager une application web
- L'utilisation de Log4J dans une servlet

Université Internationale de Casablanca

Définition d'une servlet

- **Une servlet est un programme qui s'exécute côté serveur en tant qu'extension du serveur**
- Favorise la génération de pages html dynamiques
- Elle reçoit une requête du client, elle effectue des traitements et renvoie le résultat.
- La liaison entre la servlet et le client peut être directe ou passer par un intermédiaire comme par exemple un serveur http.
- Ecrite en Java, une servlet en retire ses avantages : la portabilité, l'accès à toutes les API de Java dont JDBC pour l'accès aux bases de données, ...
- Une servlet peut être invoquée plusieurs fois en même temps pour répondre à plusieurs requêtes simultanées.

Différences entre servlets et CGI

- **Servlet vs. CGI**

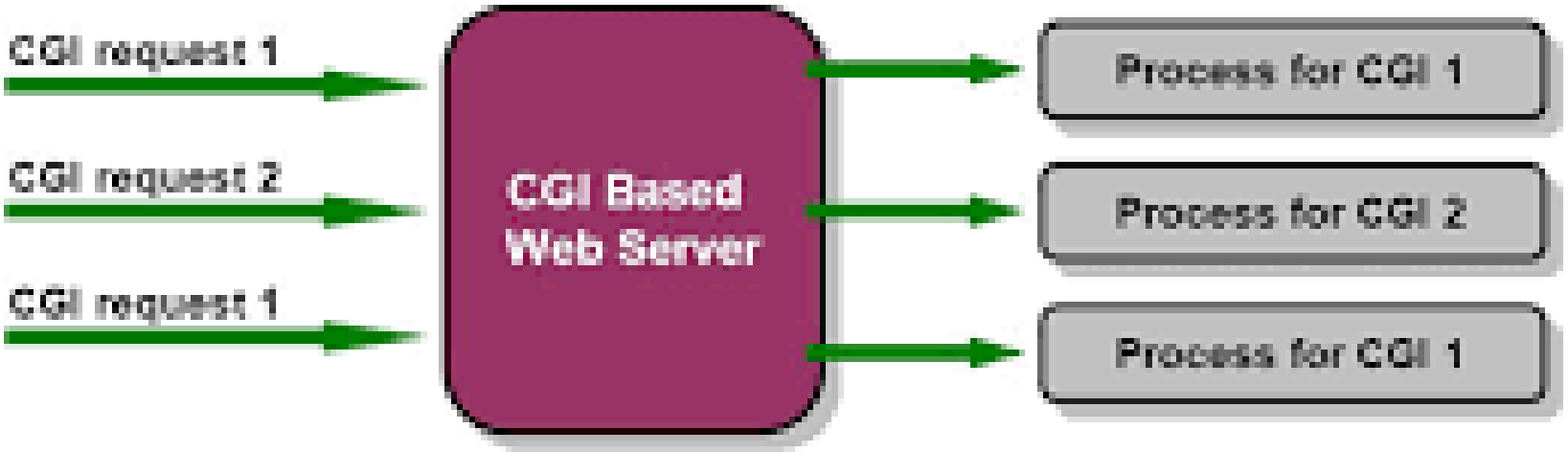
  Les programmes ou script CGI (Common Gateway Interface) sont aussi utilisés pour générer des pages HTML dynamiques. Ils représentent la plus ancienne solution pour réaliser cette tâche.

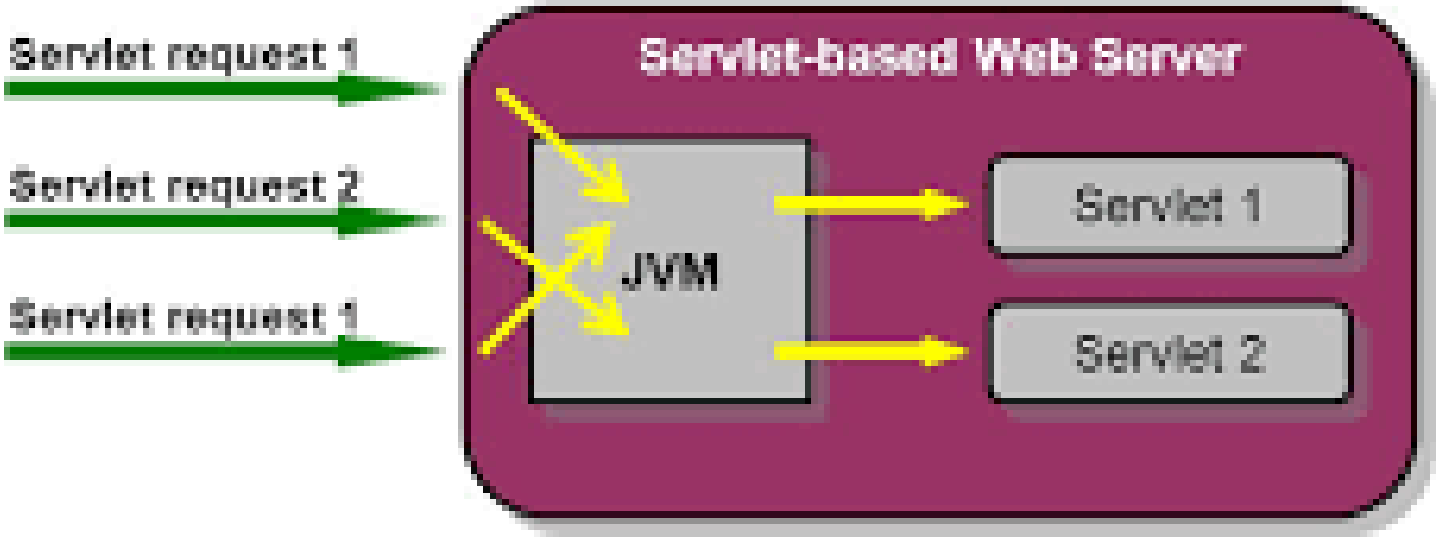
- Un CGI peut être écrit dans de nombreux langages.

- Il existe plusieurs avantages à utiliser des servlets plutôt que des CGI :

  - **Portabilité** offerte par Java bien que certains langages de script tels que PERL tournent sur plusieurs plateformes.
  - **Persistance:** la servlet reste en mémoire une fois instanciée ce qui permet de garder des ressources systèmes et gagner le temps de l'initialisation. Un CGI est chargé en mémoire à chaque requête, ce qui réduit les performances.
  - **API riches**: les servlets possèdent les avantages de toutes les classes écrites en Java : accès aux API, aux JavaBeans, le garbage collector, ...

Différences entre servlets et CGI





Université Internationale de Casablanca

Spécifications de l'API Servlets

Version	
2.0	1997
2.1	Novembre 1998, partage d'informations grâce au Servletcontext La classe GenericServlet implémente l'interface ServletConfig une méthode log() standard pour envoyer des informations dans le journal du conteneur objet RequestDispatcher pour le transfert du traitement de la requête vers une autre ressource ou inclure le résultat d'une autre ressource
2.2	Aout 1999, format war pour un déploiement standard des applications web mise en buffer de la réponse inclus dans J2EE 1.2
2.3	Septembre 2001, JSR 053 : nécessite Java 1.2 minimum ajout d'un mécanisme de filtre ajout de méthodes pour la gestion d'événements liés à la création et la destruction du context et de la session inclus dans J2EE 1.3
2.4	Novembre 2003, JSR 154 inclus dans J2EE 1.4
2.5	Septembre 2005, JSR 154 inclus dans Java EE 5, nécessite Java SE 5 minimum
3.0	Décembre 2009, JSR 315 inclus dans Java EE 6, nécessite Java SE 6 minimum
3.1	Mai 2013, JSR 340 inclus dans Java EE 7

Les méthodes HTTP

- **Les méthodes HTTP**

  Ce sont des méthodes utilisées pour des communications entre le client et le serveur.

  - **GET**: méthode utilisée pour récupérer une ressource via une URL
  - **POST**: méthode utilisée pour envoyer des données de taille volumineuse
  - **HEAD**: méthode utilisée pour envoyer seulement les en-têtes HTTP

Université Internationale de Casablanca

Requête HTTP

❑ GET

- Utilisé si on clique sur un lien ou si une adresse est saisie dans le browser
- Le corps du message n'est pas envoyé
- **On ne peut pas utiliser cette méthode pour envoyer des données volumineuses au serveur**

❑ POST

- Utilisé lorsqu'on clique sur le bouton envoyer du formulaire
- Les informations du formulaires sont contenues dans le message

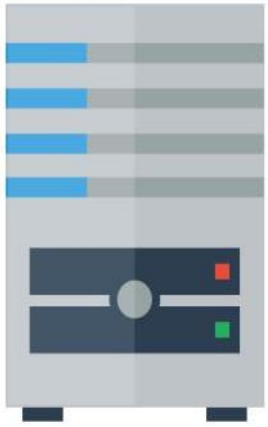
❑ HEAD

- Requêtes dont les champs d'en-tête sont renvoyés dans la réponse

Requête HTTP

REQUEST

GET/POST/HEAD http://mysitio.com →



Status Code 200 OK

← **RESPONSE**

Université Internationale de Casablanca

Code et statut d'une requête HTTP

- Le code de statut est composé de **trois chiffres** qui donnent des informations sur le résultat du traitement qui a généré cette réponse.
- Ce code peut être associé à une catégorie en fonction de sa valeur :

Plage de valeurs du code	Signification
100 à 199	Information
200 à 299	Traitement avec succès
300 à 399	La requête a été redirigée
400 à 499	La requête est incomplète ou erronée
500 à 599	Une erreur est intervenue sur le serveur

Université Internationale de Casablanca

Code et statut d'une requête HTTP

Plusieurs codes sont définis par le protocole HTTP dont les plus importants sont :

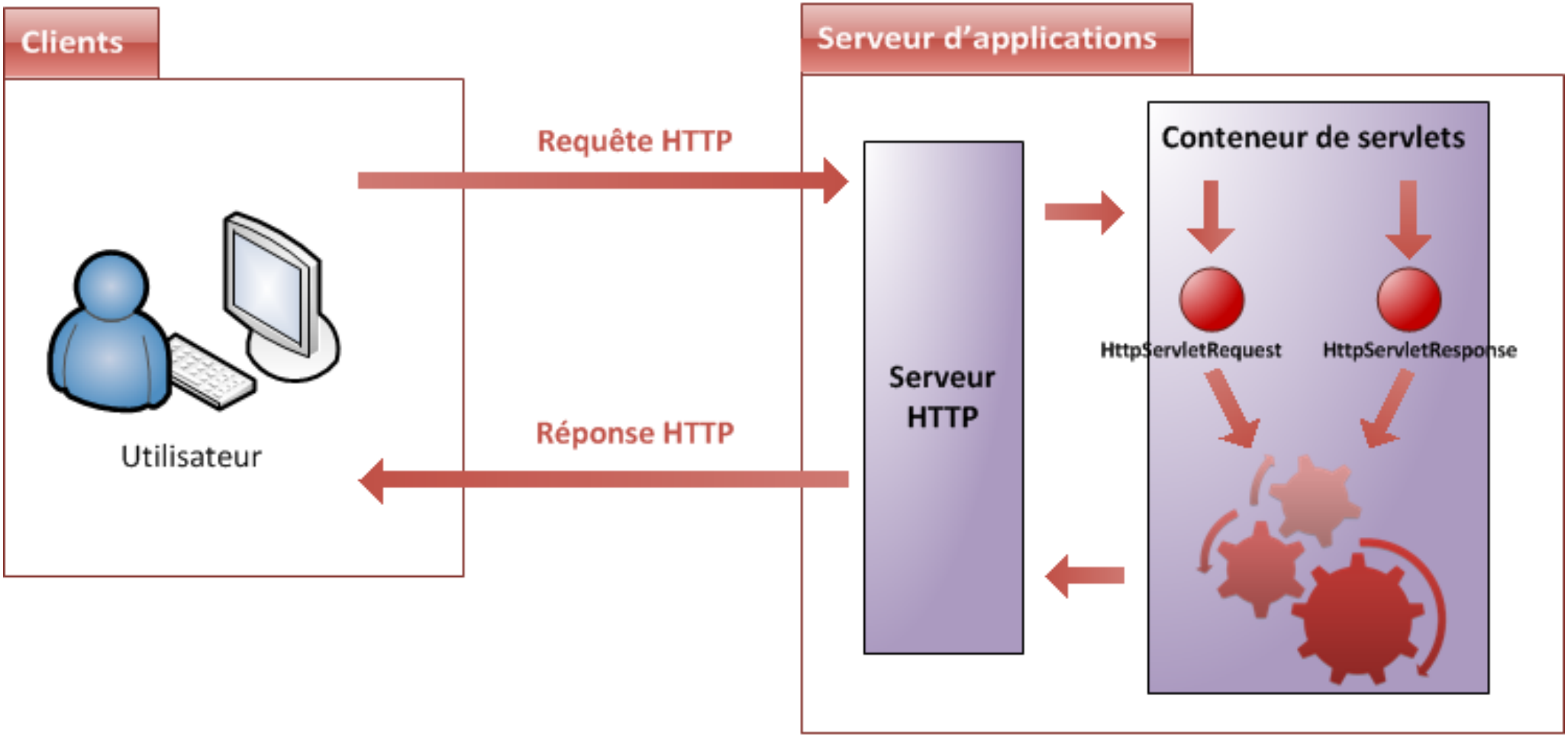
- **200** : traitement correct de la requête
- **204** : traitement correct de la requête mais la réponse ne contient aucun contenu (ceci permet au browser de laisser la page courante affichée)
- **404** : la ressource demandée n'est pas trouvée (sûrement le plus célèbre)
- **500** : erreur interne du serveur

Les méthodes implémentées par la classe HttpServlet

- Une servlet doit implémenter au moins une des méthodes **doXXX(),** afin d'être capable de traiter une requête entrante :
  - **`doGet()`** : pour gérer la méthode GET.
  - **`doPost()`**: pour gérer la méthode POST.
  - **`doHead()`**: pour gérer la méthode HEAD.

Université Internationale de Casablanca

Servlet et protocole HTTP



L'API Servlets

- L'API servlets est regroupée dans des packages préfixés par **`javax`**.
- L'API servlet regroupe un ensemble de classes dans deux packages :
  - `javax.servlet` : contient les classes pour développer des serlvets génériques indépendantes d'un protocole
  - `javax.servlet.http` : contient les classes pour développer des servlets qui reposent sur le protocole http utilisé par les serveurs web.

Université Internationale de Casablanca

Le package javax.servlet

javax.servlet	Nom	Rôle
Les interfaces	RequestDispatcher	Définition d'un objet qui permet le renvoi d'une requête vers une autre ressource du serveur (une autre servlet, une JSP ...)
	Servlet	Définition de base d'une servlet
	ServletConfig	Définition d'un objet pour configurer la servlet
	ServletContext	Définition d'un objet pour obtenir des informations sur le contexte d'exécution de la servlet
	ServletRequest	Définition d'un objet contenant la requête du client
	ServletResponse	Définition d'un objet qui contient la réponse renvoyée par la servlet
	SingleThreadModel	Permet de définir une servlet qui ne répondra qu'à une seule requête à la fois
Les classes	GenericServlet	Classe définissant une servlet indépendante de tout protocole
	ServletInputStream	Flux permettant la lecture des données de la requête cliente
	ServletOutPutStream	Flux permettant l'envoi de la réponse de la servlet
Les exceptions	SevletException	Exception générale en cas de problème durant l'exécution de la servlet
	UnavailableException	Exception levée si la servlet n'est pas disponible

Université Internationale de Casablanca

Le package javax.servlet.http

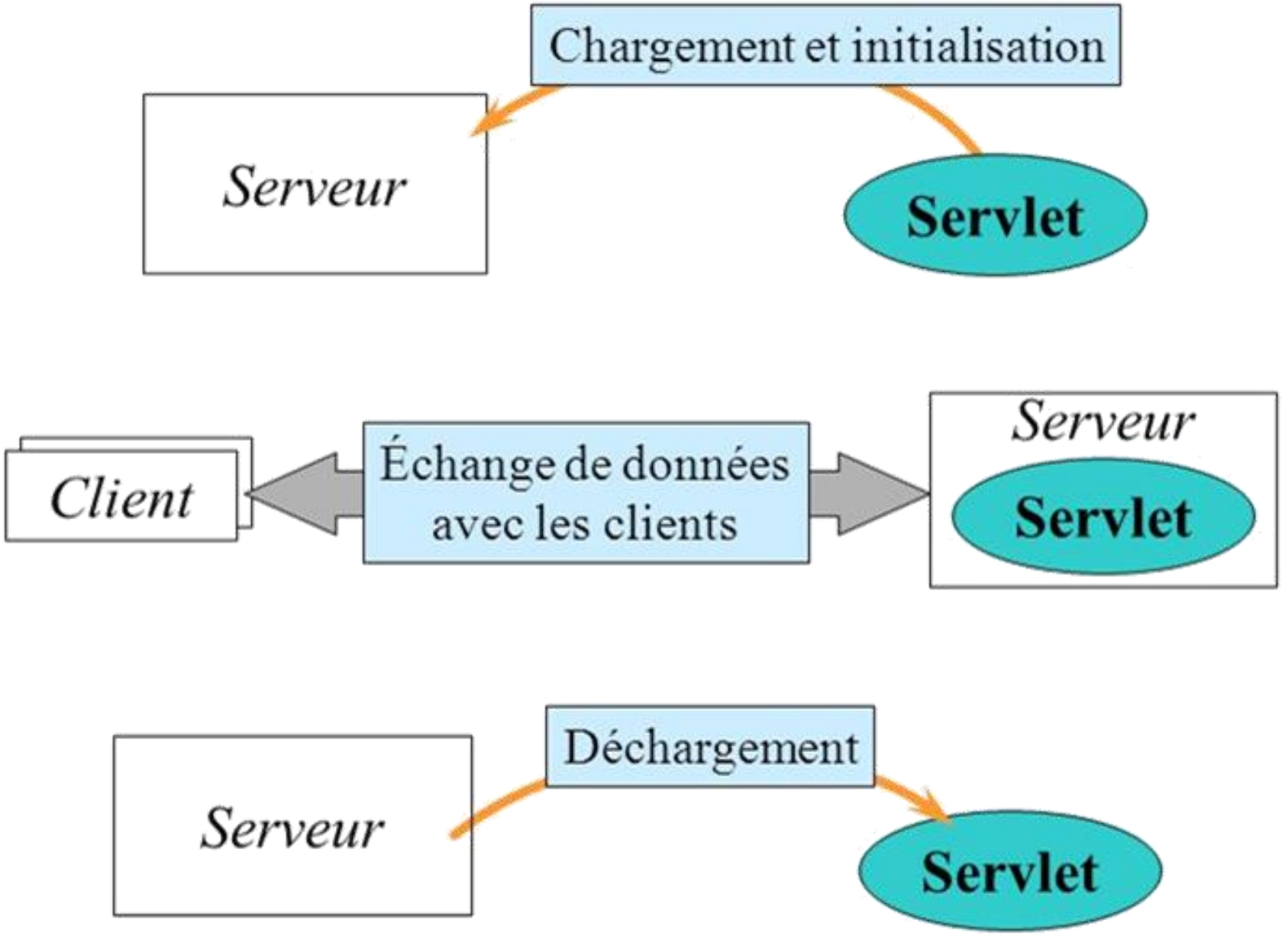
Javax.servlet	Nom	Rôle
Les interfaces	HttpServletRequest	Hérite de ServletRequest : définit un objet contenant une requête selon le protocole http
	HttpServletResponse	Hérite de ServletResponse : définit un objet contenant la réponse de la servlet selon le protocole http
	HttpSession	Définit un objet qui représente une session
Les classes	Cookie	Classe représentant un cookie (ensemble de données sauvegardées par le brower sur le poste client)
	HttpServlet	Hérite de GenericServlet : classe définissant une servlet utilisant le protocole http
	HttpUtils	Classe proposant des méthodes statiques utiles pour le développement de servlets http

Université Internationale de Casablanca

La classe HttpServlet

- Une servlet est une simple classe Java, qui a la particularité de permettre le traitement de requêtes et la personnalisation de réponses.
- C'est l'interface mère que toute servlet doit obligatoirement implémenter.
- Elle encapsule deux objets fondamentaux:
  - **`HttpServetRequest`** : cet objet contient la requête HTTP, et donne accès à toutes ses informations, telles que les en-têtes (headers) et le corps de la requête.
  - **`HttpServletResponse`**: cet objet initialise la réponse HTTP qui sera renvoyée au client, et permet de la personnaliser, en initialisant par exemple les en-têtes et le corps.

Cycle de vie d'une servlet



Université Internationale de Casablanca

Méthodes associées au cycle de vie d'une servlet

Méthode	Rôle
void service (ServletRequest req, ServletResponse res)	Cette méthode est exécutée par le conteneur lorsque la servlet est sollicitée : chaque requête du client déclenche une seule exécution de cette méthode. Cette méthode pouvant être exécutée par plusieurs threads, il faut prévoir un processus d'exclusion pour l'utilisation de certaines ressources.
void init(ServletConfig conf)	Initialisation de la servlet. Cette méthode est appelée une seule fois après l'instanciation de la servlet. Aucun traitement ne peut être effectué par la servlet tant que l'exécution de cette méthode n'est pas terminée.
void destroy()	Cette méthode est appelée lors de la destruction de la servlet. Elle permet de libérer proprement certaines ressources (fichiers, bases de données ...). C'est le serveur qui appelle cette méthode.

L'interface ServletRequest

L'interface ServletRequest définit plusieurs méthodes qui permettent d'obtenir des données sur la requête du client

Méthode	Rôle
ServletInputStream getInputStream()	Permet d'obtenir un flux pour les données de la requête
BufferedReader getReader()	Idem

L'interface ServletResponse

L'interface ServletResponse définit plusieurs méthodes qui permettent de fournir la réponse faite par la servlet suite à ses traitements :

Méthode	Rôle
SetContentType	Permet de préciser le type MIME de la réponse
ServletOutputStream getOutputStream()	Permet d'obtenir un flux pour envoyer la réponse
PrintWriter getWriter()	Permet d'obtenir un flux pour envoyer la réponse

Université Internationale de Casablanca

Un exemple de Servlet

Exemple (code Java 1.1) :

```
1  import java.io.*;
2  import javax.servlet.*;
3
4  public class TestServlet implements Servlet {
5    private ServletConfig cfg;
6
7    public void init(ServletConfig config) throws ServletException {
8       cfg = config;
9    }
10
11   public ServletConfig getServletConfig() {
12     return cfg;
13   }
14
15   public String getServletInfo() {
16     return "Une servlet de test";
17   }
18
19   public void destroy() {
20   }
21
22   public void service (ServletRequest req,  ServletResponse res )
23   throws ServletException, IOException  {
24     res.setContentType( "text/html" );
25     PrintWriter out = res.getWriter();
26     out.println( "<HTML>" );
27     out.println( "<HEAD>");
28     out.println( "<TITLE>Page generee par une servlet</TITLE>" );
29     out.println( "</HEAD>" );
30     out.println( "<BODY>" );
31     out.println( "<H1>Bonjour</H1>" );
32     out.println( "</BODY>" );
33     out.println( "</HTML>" );
34     out.close();
35   }
36 }
```

Les méthodes doGet()

- **Une requête de type GET est utile avec des liens**

  Le traitement typique de la méthode doGet() est d'analyser les paramètres de la requête, alimenter les données de l'en-tête de la réponse et d'écrire la réponse.

La signature de la méthode doGet() :

```
1  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
2  }
```

Université Internationale de Casablanca

Les méthodes doPost()

- **Une requête de type POST n'est utile qu'avec un formulaire HTML**

Exemple : de code HTML

```
1  <FORM ACTION="http://localhost:8080/examples/servlet/tomcat1.TestPostServlet"
2  METHOD="POST">
3  <INPUT NAME="NOM">
4  <INPUT NAME="PRENOM">
5  <INPUT TYPE="ENVOYER">
6  </FORM>
```

La signature de la méthode doPost() :

```
1  protected void doPost(HttpServletRequest request, HttpServletResponse response)
2  throws IOException {
3  }
```

Les méthodes doPost()

- **Utiliser la méthode getParameter() de l'objet HttpServletRequest pour obtenir la valeur associée à chaque paramètre .**

Exemple :

```
1  public void doPost(HttpServletRequest request, HttpServletResponse response)
2  throws IOException, ServletException {
3      String nom = request.getParameter("NOM");
4      String prenom = request.getParameter("PRENOM");
5  }
```

La génération de la réponse

- **La servlet envoie sa réponse au client en utilisant un objet de type HttpServetResponse.**

Méthode	Rôle
void sendError (int)	Envoie une erreur avec un code retour et un message par défaut
void sendError (int, String)	Envoie une erreur avec un code retour et un message
void setContentType(String)	Héritée de ServletResponse, cette méthode permet de préciser le type MIME de la réponse
void setContentLength(int)	Héritée de ServletResponse, cette méthode permet de préciser la longueur de la réponse
ServletOutputStream getOutputStream()	Héritée de ServletResponse, elle retourne un flux pour l'envoi de la réponse
PrintWriter getWriter()	Héritée de ServletResponse, elle retourne un flux pour l'envoi de la réponse

Université Internationale de Casablanca

La génération de la réponse via StringBuffer et GetOutputStream

Exemple (code Java 1.1) :

```
1  protected void GenererReponse1(HttpServletResponse reponse) throws IOException {
2     //creation de la reponse
3     StringBuffer sb = new StringBuffer();
4     sb.append("<HTML>\n");
5     sb.append("<HEAD>\n");
6     sb.append("<TITLE>Bonjour</TITLE>\n");
7     sb.append("</HEAD>\n");
8     sb.append("<BODY>\n");
9     sb.append("<H1>Bonjour</H1>\n");
10    sb.append("</BODY>\n");
11    sb.append("</HTML>");
12
13    // envoi des infos de l'en-tete
14    reponse.setContentType("text/html");
15    reponse.setContentLength(sb.length());
16
17    // envoi de la réponse
18    reponse.getOutputStream().print(sb.toString());
19 }
```

Université Internationale de Casablanca

La génération de la réponse via GetOutputStream

Exemple :

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4
5  public class TestServlet4 extends HttpServlet {
6
7     public void doGet(HttpServletRequest req, HttpServletResponse res)
8        throws ServletException, IOException {
9        res.setContentType("text/html");
10       ServletOutputStream out = res.getOutputStream();
11       out.println("<HTML>\n");
12       out.println("<HEAD>\n");
13       out.println("<TITLE>Bonjour</TITLE>\n");
14       out.println("</HEAD>\n");
15       out.println("<BODY>\n");
16       out.println("<H1>Bonjour</H1>\n");
17       out.println("</BODY>\n");
18       out.println("</HTML>");
19    }
20 }
```

Université Internationale de Casablanca

La génération de la réponse via la méthode GetWriter()

Exemple (code Java 1.1) :

```
1  protected void GenererReponse2(HttpServletResponse reponse) throws IOException {
2
3      reponse.setContentType("text/html");
4
5      PrintWriter out = reponse.getWriter();
6
7      out.println("<HTML>");
8      out.println("<HEAD>");
9      out.println("<TITLE>Bonjour</TITLE>");
10     out.println("</HEAD>");
11     out.println("<BODY>");
12     out.println("<H1>Bonjour</H1>");
13     out.println("</BODY>");
14     out.println("</HTML>");
15 }
```

Université Internationale de Casablanca

Mise en place d'une Servlet

- Il faut déclarer une nouvelle classe qui hérite de HttpServlet.
- Il faut redéfinir la méthode doGet() pour y insérer le code qui va envoyer dans un flux le code HTML de la page générée.
- Toute servlet doit au moins importer trois packages : **`java.io`** pour la gestion des flux et deux packages de l'API servlet :
  - **`javax.servlet.*`**
  - **`javax.servlet.http`**

Université Internationale de Casablanca

Exemple de servlet HTTP très simple

Exemple (code Java 1.1) :

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4
5  public class MyHelloServlet extends HttpServlet {
6
7    public void doGet(HttpServletRequest request, HttpServletResponse response)
8      throws IOException, ServletException {
9      response.setContentType("text/html");
10
11     PrintWriter out = response.getWriter();
12
13     out.println("<html>");
14     out.println("<head>");
15     out.println("<title>Bonjour tout le monde</title>");
16     out.println("</head>");
17     out.println("<body>");
18     out.println("<h1>Bonjour tout le monde</h1>");
19     out.println("</body>");
20     out.println("</html>");
21   }
22 }
```

Les paramètres d'initialisation

ServletConfig est une interface qui possède deux méthodes permettant de connaître les paramètres d'initialisation :

- `String getInitParameter(String)` : retourne la valeur du paramètre dont le nom est fourni en paramètre
- `Enumeration getInitParameterNames()` : retourne une énumération des paramètres d'initialisation

Les paramètres d'initialisation

Exemple :

```
1  String param;
2
3  public void init(ServletConfig config) {
4    param = config.getInitParameter("param");
5  }
```

Exemple (code Java 1.1) :

```
1  public void init(ServletConfig config) throws ServletException {
2
3    cfg = config;
4
5    System.out.println("Liste des parametres d'initialisation");
6
7    for (Enumeration e=config.getInitParameterNames(); e.hasMoreElements();) {
8      System.out.println(e.nextElement());
9    }
10 }
```

Université Internationale de Casablanca

Gestion du fichier log du serveur

La servlet peut obtenir des informations à partir d'un objet **ServletContext** retourné par la méthode **getServletContext()** d'un objet ServletConfig.

méthode	Rôle	Deprecated
String getMimeType(String)	Retourne le type MIME du fichier en paramètre	
String getServletInfo()	Retourne le nom et le numéro de version du moteur de servlet	
Servlet getServlet(String)	Retourne une servlet à partir de son nom grâce au context	Ne plus utiliser depuis la version 2.1 du jsdk
Enumeration getServletNames()	Retourne une énumération qui contient la liste des servlets relatives au contexte	Ne plus utiliser depuis la version 2.1 du jsdk
void log(Exception, String)	Ecrit les informations fournies en paramètre dans le fichier log du serveur	Utiliser la nouvelle méthode surchargée log()
void log(String)	Idem	
void log (String, Throwable)	Idem	

Université Internationale de Casablanca

Ecriture dans le fichier log du serveur

Exemple : écriture dans le fichier log du serveur

```
1  public void init(ServletConfig config) throws ServletException {
2    ServletContext sc = config.getServletContext();
3    sc.log( "Demarrage servlet TestServlet" );
4  }
```

Le format du fichier log est dépendant du serveur

Exemple : résultat avec tomcat

```
1  Context log path="/examples" :Demarrage servlet TestServlet
```

Les informations contenues dans une requête

- L’informations en provenance du client peuvent être extraites de l'objet ServletRequest
- Les informations les plus utiles sont les paramètres envoyés dans la requête.

Université Internationale de Casablanca

Les informations contenues dans une requête

Exemple (code Java 1.1) :

```
1  package tomcat1;
2
3  import java.io.*;
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.util.*;
7
8  public class InfoServlet extends HttpServlet {
9
10   public void doGet(HttpServletRequest request, HttpServletResponse response)
11     throws IOException,ServletException {
12     GenererReponse(request, response);
13   }
14
15   protected void GenererReponse(HttpServletRequest request, HttpServletResponse reponse)
16     throws IOException {
17
18     reponse.setContentType("text/html");
19
20     PrintWriter out =reponse.getWriter();
```

Université Internationale de Casablanca

Les informations contenues dans une requête

```
22      out.println("<html>");
23      out.println("<body>");
24      out.println("<head>");
25      out.println("<title>Informations a disposition de la servlet</title>");
26      out.println("</head>");
27      out.println("<body>");
28      out.println("<p>Type mime de la requête :"
29        +request.getContentType()+"</p>");
30      out.println("<p>Protocole de la requête :"
31        +request.getProtocol()+"</p>");
32      out.println("<p>Adresse IP du client :"
33        +request.getRemoteAddr()+"</p>");
34      out.println("<p>Nom du client : "
35        +request.getRemoteHost()+"</p>");
36      out.println("<p>Nom du serveur qui a reçu la requête :"
37        +request.getServerName()+"</p>");
38      out.println("<p>Port du serveur qui a reçu la requête :"
39        +request.getServerPort()+"</p>");
40      out.println("<p>scheme: "+request.getScheme()+"</p>");
41      out.println("<p>liste des paramètres </p>");
```

Université Internationale de Casablanca

Les informations contenues dans une requête

```
43      for (Enumeration e =request.getParameterNames() ; e.hasMoreElements() ; ) {
44        Object p = e.nextElement();
45        out.println("<p>  nom : "+p+" valeur :"
46          +request.getParameter(""+p)+"</p>");
47      }
48
49      out.println("</body>");
50      out.println("</html>");
51    }
52  }
```

Université Internationale de Casablanca

Les informations contenues dans une requête

Résultat : avec l'url http://localhost:8080/examples/servlet/tomcat1.InfoServlet?param1=valeur1&param2=valeur2 :
Une page html s'affiche contenant :

```
1  Type mime de la requête : null
2
3  Protocole de la requête : HTTP/1.0
4
5  Adresse IP du client : 127.0.0.1
6
7  Nom du client : localhost
8
9  Nom du serveur qui a reçu la requête : localhost
10
11 Port du serveur qui a reçu la requête : 8080
12
13 scheme : http
14
15 liste des paramètres
16
17    nom : param2 valeur :valeur2
18
19    nom : param1 valeur :valeur1
```

Les informations contenues dans une requête

- L'interface **ServletRequest** dispose de nombreuses méthodes pour obtenir ces informations :

Méthode	Rôle
int getContentLength()	Renvoie la taille de la requête, 0 si elle est inconnue
String getContentType()	Renvoie le type MIME de la requête, null s'il est inconnu
ServletInputStream getInputStream()	Renvoie un flux qui contient le corps de la requête
Enumeration getParameterNames()	Renvoie une énumération contenant le nom de tous les paramètres
String getProtocol()	Retourne le nom du protocole utilisé par la requête et sa version
BufferedReader getReader()	Renvoie un flux qui contient le corps de la requête
String getRemoteAddr()	Renvoie l'adresse IP du client
String getRemoteHost()	Renvoie le nom de la machine cliente
String getScheme	Renvoie le protocole utilisé par la requête (exemple : http, ftp ...)
String getServerName()	Renvoie le nom du serveur qui a reçu la requête
int getServerPort()	Renvoie le port du serveur qui a reçu la requête

Université Internationale de Casablanca

Déclaration d'une Servlet dans le web.xml

- Déclaration dans le fichier de configuration **web.xml**
- Les balises responsables de la définition d'une servlet:
  - **`<servlet-name>`**: permet de donner un nom à la servlet
  - **`<servlet-class>`**: sert à préciser le chemin de la classe de la servlet dans l'application
  - **`<description>`**: permet de décrire plus amplement le rôle de la servlet
  - **`<init-param>`**: permet de préciser des paramètres qui seront accessibles à la servlet lors de son chargement
  - **`<load-on-startup>`**: permet de forcer le chargement de la servlet dès le démarrage du serveur

Déclaration d'une Servlet dans le web.xml

xml

```xml
<servlet>
    <servlet-name>Test</servlet-name>
    <servlet-class>com.sdzee.servlets.Test</servlet-class>

    <description>Ma première servlet de test.</description

    <init-param>
        <param-name>auteur</param-name>
        <param-value>Coyote</param-value>
    </init-param>

    <load-on-startup>1</load-on-startup>
</servlet>
```

Université Internationale de Casablanca

Mapping d'une Servlet

- Faire correspondre notre servlet fraîchement déclarée à une URL
- La balises responsable du mapping d'une servlet `<servlet-mapping>`:
  - `<servlet-name>`: permet de préciser le nom de la servlet à laquelle faire référence
  - `<url-pattern>`: permet de préciser la ou les URL relatives au travers desquelles la servlet sera accessible

- **L'ordre des sections de déclaration au sein du fichier est important : il est impératif de définir une servlet avant de spécifier son mapping.**

Mapping d'une Servlet

xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
    <servlet>
        <servlet-name>Test</servlet-name>
        <servlet-class>com.sdzee.servlets.Test</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>Test</servlet-name>
        <url-pattern>/toto</url-pattern>
    </servlet-mapping>
</web-app>
```

Surcharge d'une Servlet

```
1  package com.sdzee.servlets;
2
3  import java.io.IOException;
4
5  import javax.servlet.ServletException;
6  import javax.servlet.http.HttpServlet;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 public class Test extends HttpServlet {
11     public void doGet( HttpServletRequest request, HttpServletResponse response ) throws
   ServletException, IOException{
12
13     }
14 }
```

Université Internationale de Casablanca

Envoi d'une réponse HTML au client

```
1  public void doGet( HttpServletRequest request, HttpServletResponse response ) throws ServletException,
   IOException{
2      response.setContentType("text/html");
3      response.setCharacterEncoding( "UTF-8" );
4      PrintWriter out = response.getWriter();
5      out.println("<!DOCTYPE html>");
6      out.println("<html>");
7      out.println("<head>");
8      out.println("<meta charset=\"utf-8\" />");
9      out.println("<title>Test</title>");
10     out.println("</head>");
11     out.println("<body>");
12     out.println("<p>Ceci est une page générée depuis une servlet.</p>");
13     out.println("</body>");
14     out.println("</html>");
15 }
```

Notification d'une réponse HTML

Modification de l'encodage

Permet l'envoi du texte au client

Formatage HTML

Université Internationale de Casablanca

Résumé sur les Servlets

- Le client envoie des requêtes au serveur grâce aux méthodes du protocole HTTP, notamment GET, POST et HEAD.
- Le conteneur web place chaque requête reçue dans un objet **HttpServletRequest**, et place chaque réponse qu'il initialise dans l'objet **HttpServletResponse**.
- Le conteneur transmet chaque couple requête/réponse à une servlet : c'est un objet Java assigné à une requête et capable de générer une réponse en conséquence.
- La servlet est donc le point d'entrée d'une application web, et se déclare dans son fichier de configuration web.xml.
- Une servlet peut se charger de répondre à une requête en particulier, ou à un groupe entier de requêtes.
- Pour pouvoir traiter une requête HTTP de type GET, une servlet doit implémenter la méthode **doGet()** ; pour répondre à une requête de type POST, la méthode **doPost()** ; etc.

QUESTIONS ?