Les Servlets

Rappels : l'architecture client serveur du web

(Merci à Pascal Graffion pour cette partie)

Le navigateur Web

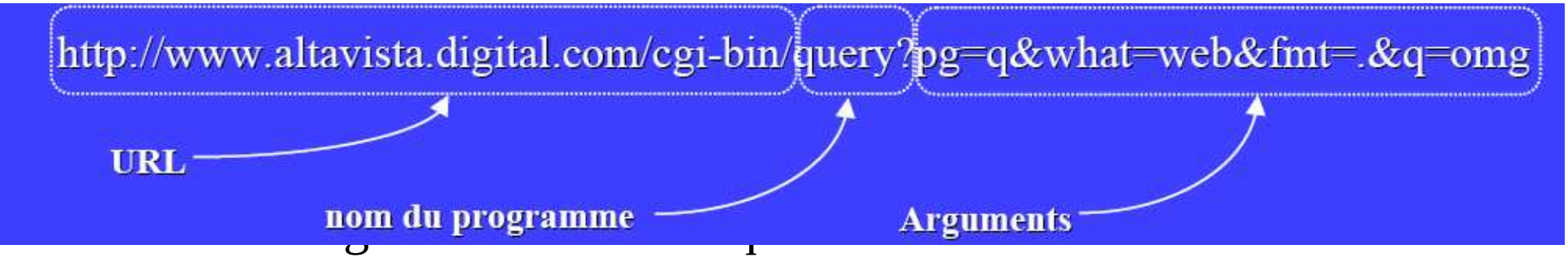
- Il est installé sur le poste client
- Il affiche des pages web
- Il sollicite le serveur web par une requête et reçoit une réponse
- Il communique avec le serveur Web par le protocole HTTP

Le serveur Web

- C’est un programme situé sur une machine connectée au réseau internet
- Il détecte les requêtes des clients pour les traiter dans l’ordre de leur arrivée
- Il renvoie une page HTML
- Il communique avec le client avec le protocole HTTP

Communication serveur Web / navigateur Web (1)

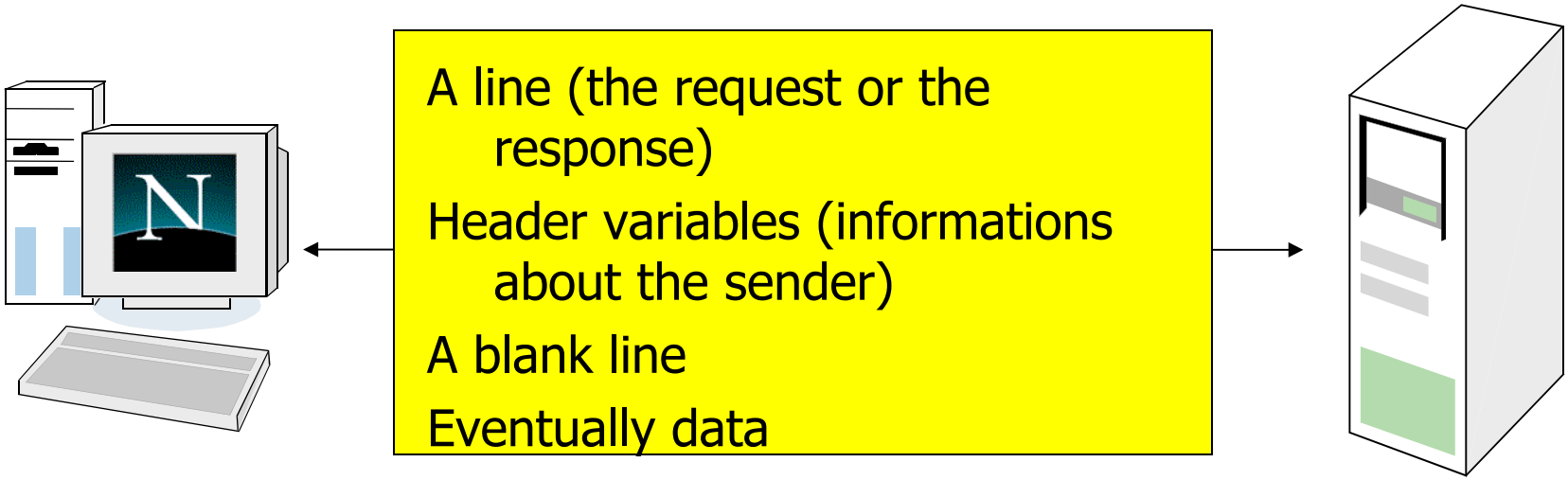
- Le navigateur formule une requête HTTP référençant un document :
  - statique : http://cedric.cnam.fr/~farinone/Java/index.html
  - ou dynamique :



- Le serveur traite la requête et renvoie la réponse

HTTP

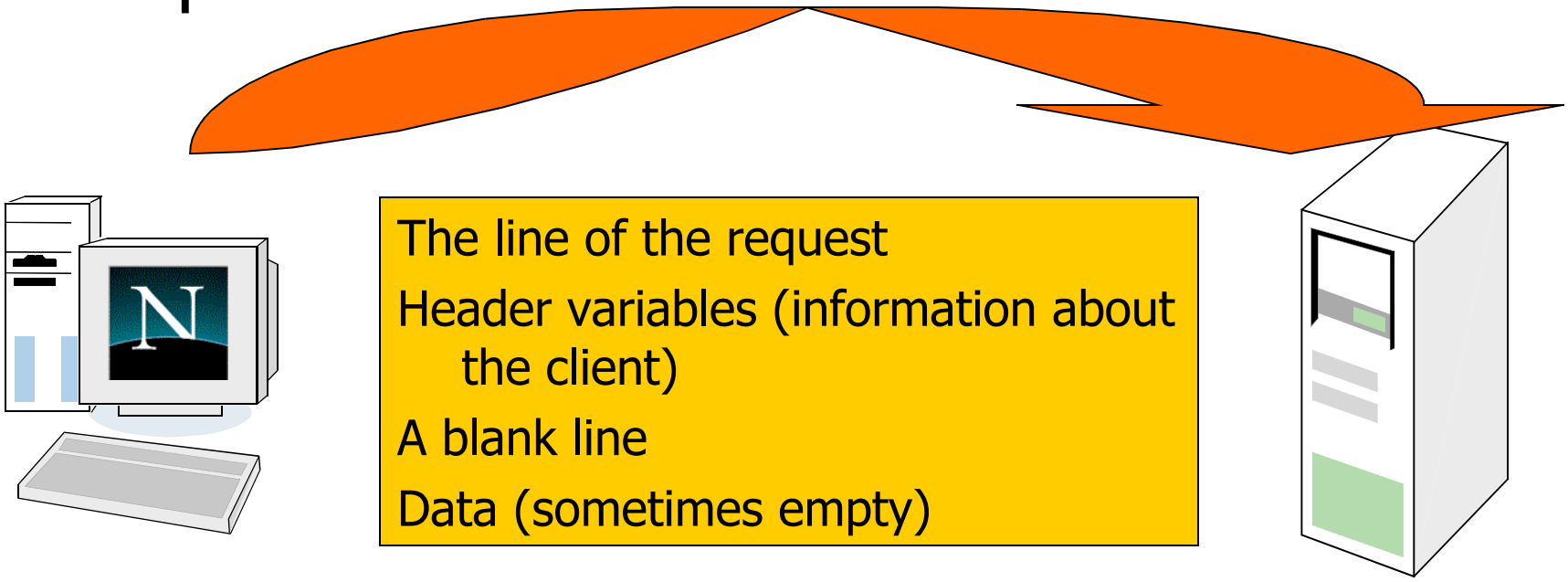
- A protocol to obtain resources (not only HTML pages)
- Format of the exchanged messages



- A text-oriented protocol (maybe except the data)

HTTP: a request (1/2)

- A message from client to server = a request



The line of the request

Header variables (information about the client)

A blank line

Data (sometimes empty)

HTTP: a request (2/2)

`http://service:8080/index.html`

=

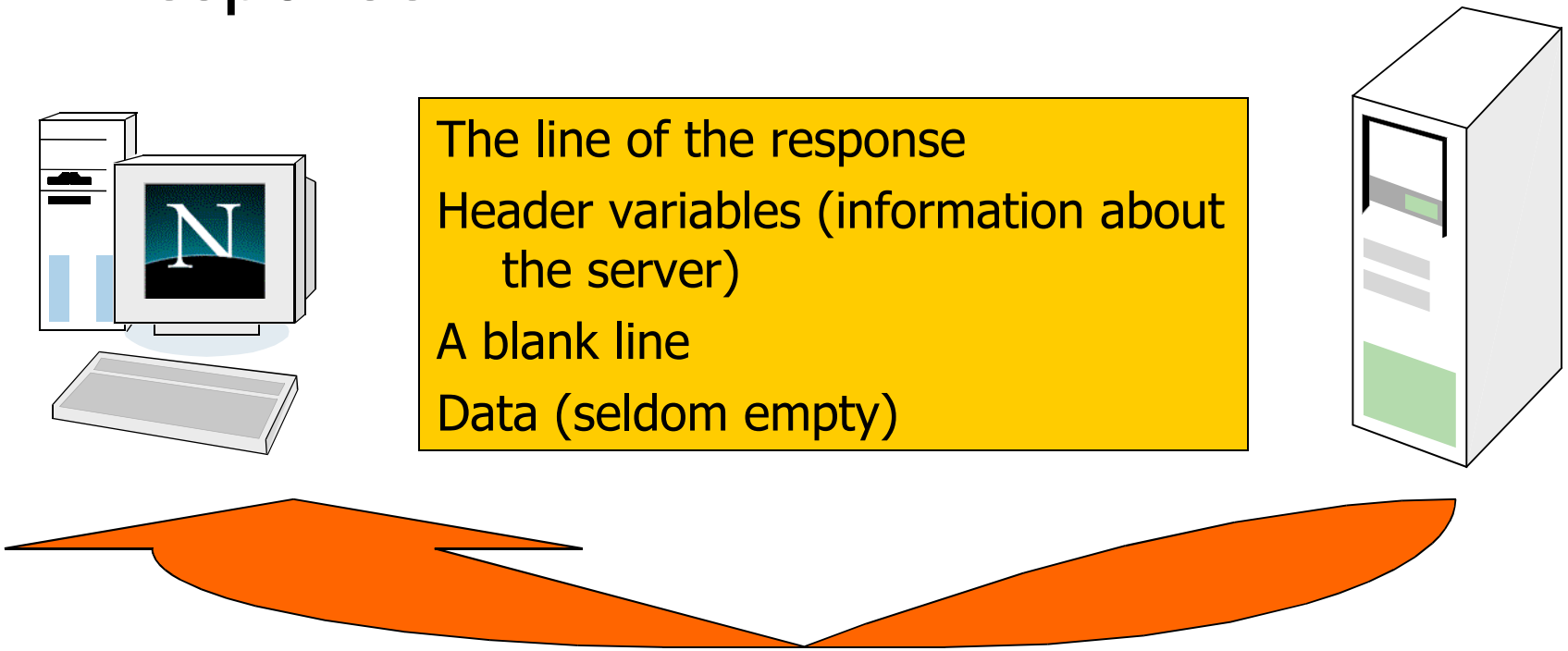
1°) Open a TCP connection to `service` on port `8080`

2°) Send a HTTP message (a request) which first line is:

```
get /index.html HTTP/1.1
```

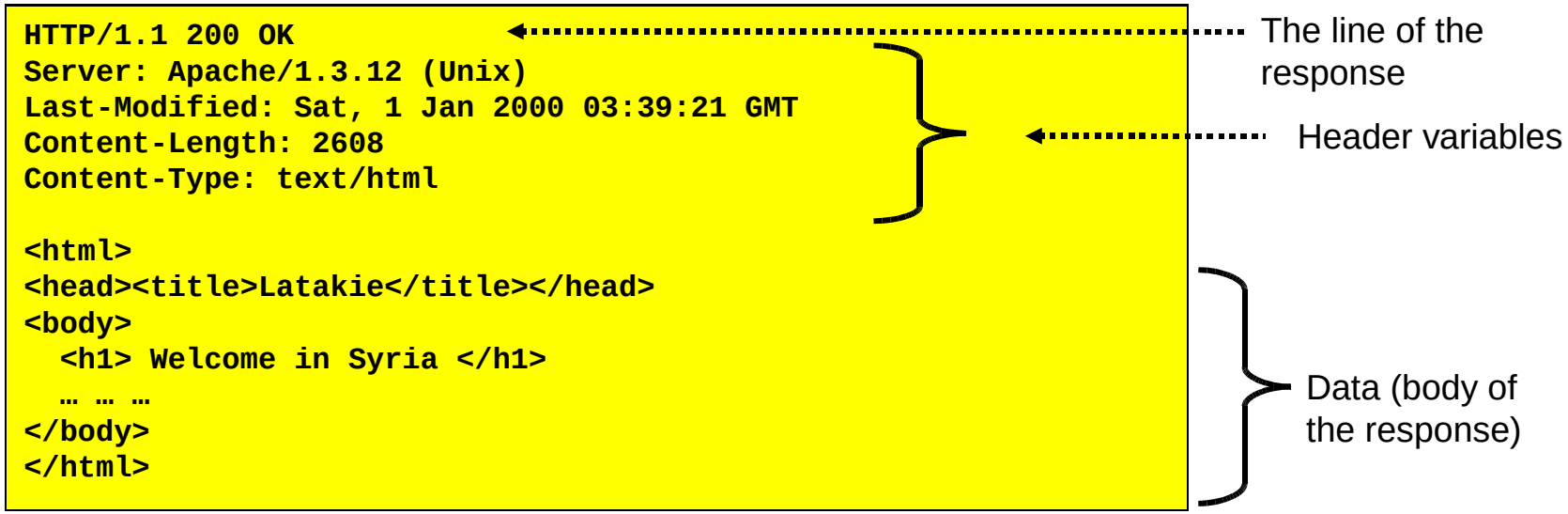
HTTP: a response (1/2)

- A message from server to client = a response



HTTP: a response (2/2)

- Example of response:



Traitement d’une requête HTTP

Le serveur web détermine si une requête est dynamique en fonction :

- du répertoire quelle référence (cgi-bin/, servlet/, ...)
- ou de l'extension du fichier référencé (. jsp, .php, ...)

Délègue l’exécution de la requête à :

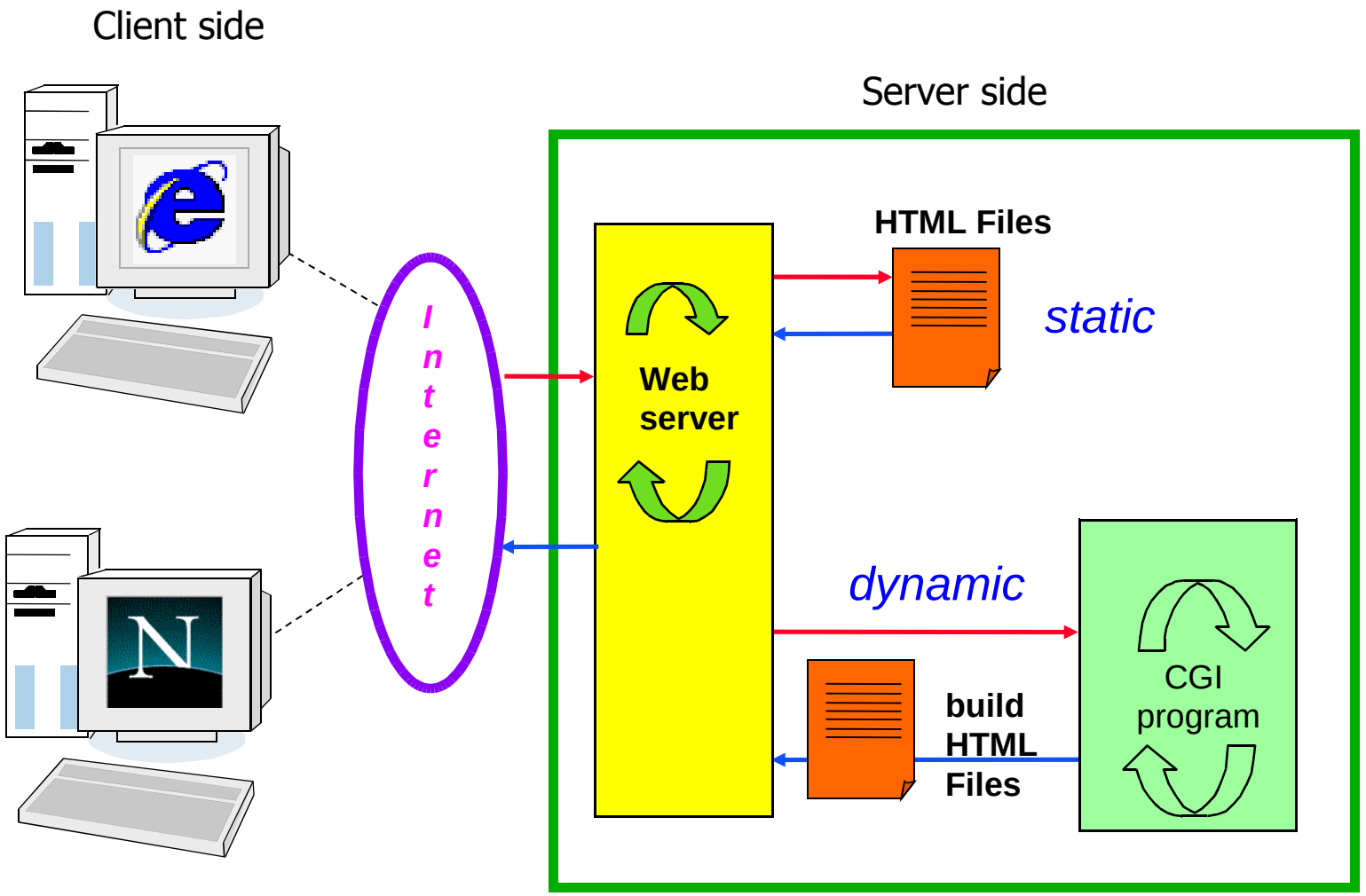
- une JVM pour une servlet Java
- un interpréteur adéquat pour php, asp, jsp, ...

Techniques utilisées côté serveur

Les requêtes peuvent être traitées par différentes technologies :

– CGI (Common Gateway Interface)
– API de serveur WEB propriétaires (ISAPI, NSAPI)
– Servlets Java
– JSP (JavaServer Pages)
– ASP
– PHP
– ...

Static and dynamic web: conclusion

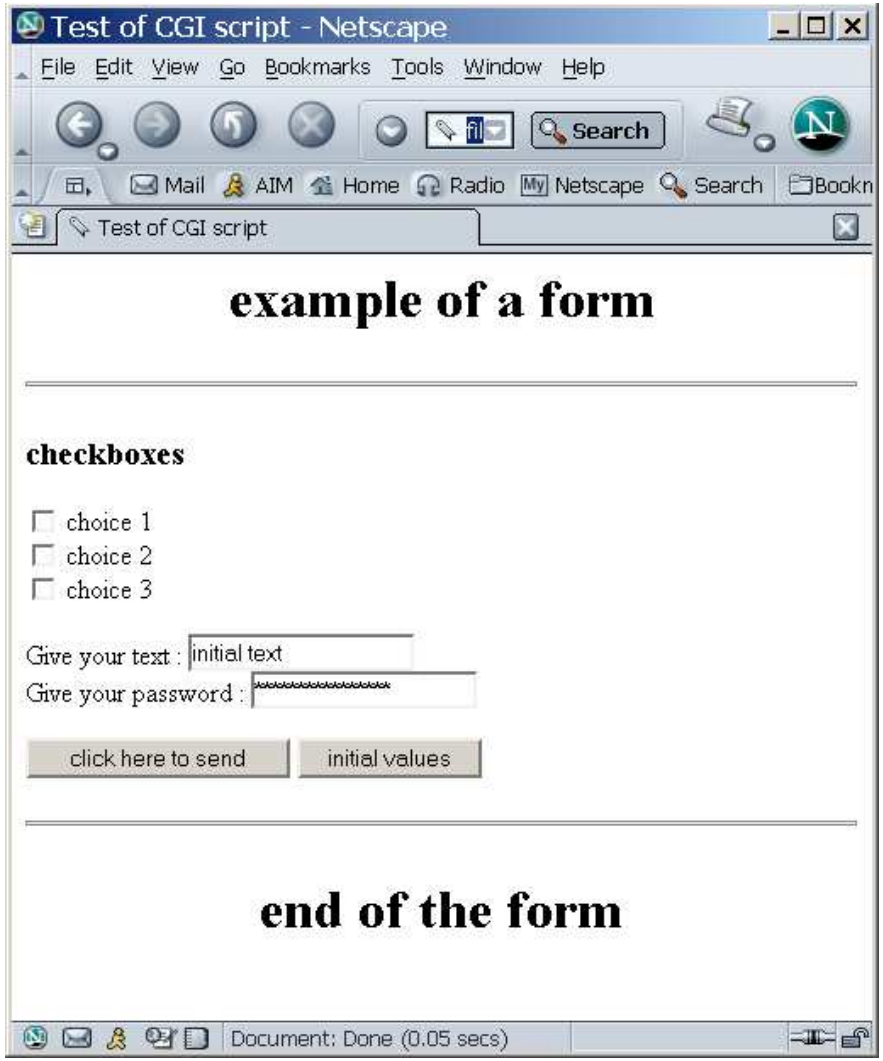


Parameters of CGI Programs

- Generally these programs are launched with parameters
- Give examples. Who first gives these parameters ?

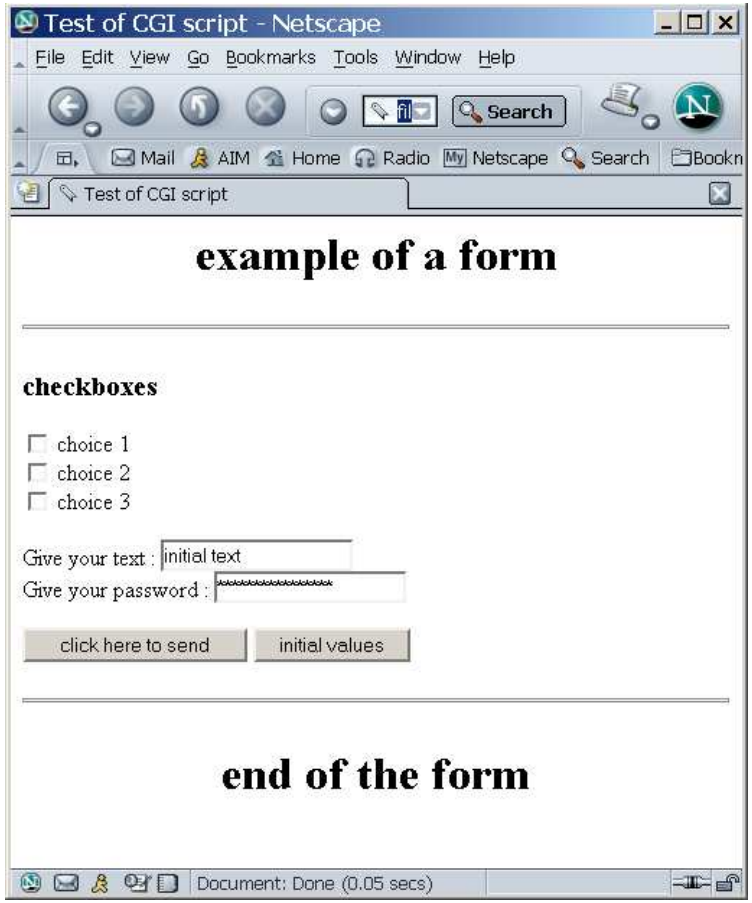
- So HTML (2.0) proposes GUI to launch programs and give them parameters : the form
- GUI can have :
  - checkboxes, textfield, textfield with * as echo character, radio button, reset button, button to launch the request, selection list, textarea

GUI with a form



The code of the form

- The form:



is obtained by:

```
<html>
<head>
   <title>Test of CGI script</title>
</head>
<body>
<h1 ALIGN=CENTER>example of a form</h1>
<hr>

<form METHOD="POST" ACTION="http://localhost:8080/">
<h3>checkboxes</h3>
<input TYPE="…
<!-- the rest is the … exercise -->
```

So the parameters to the program?

- The GUI components have a name. This name is used by the program. The program receives a line constitued by a `NameOfComponent=value` separated by &
- Some transformations (blank character, ...)

```
<INPUT TYPE="CHECKBOX" NAME="choice1" VALUE="choice 1 is selected">the choice 1<BR>
<INPUT TYPE="CHECKBOX" NAME="choice2">second choice<BR>
<INPUT TYPE="CHECKBOX" NAME="choice3">and the third one<BR>
<P>
Give your text : <INPUT TYPE="TEXT" NAME="wtext" VALUE="initial text"><BR>
Give your password : <INPUT TYPE="PASSWORD" NAME="wpass" VALUE="initial password"><BR>
<P>
<INPUT TYPE="SUBMIT" NAME="Send" VALUE="click here to send">
```

- is sent `choice1=choice+1+is+selected&choice2=on&wtext=my+text&wpass=my+passwd&Send=click+here+to+send`

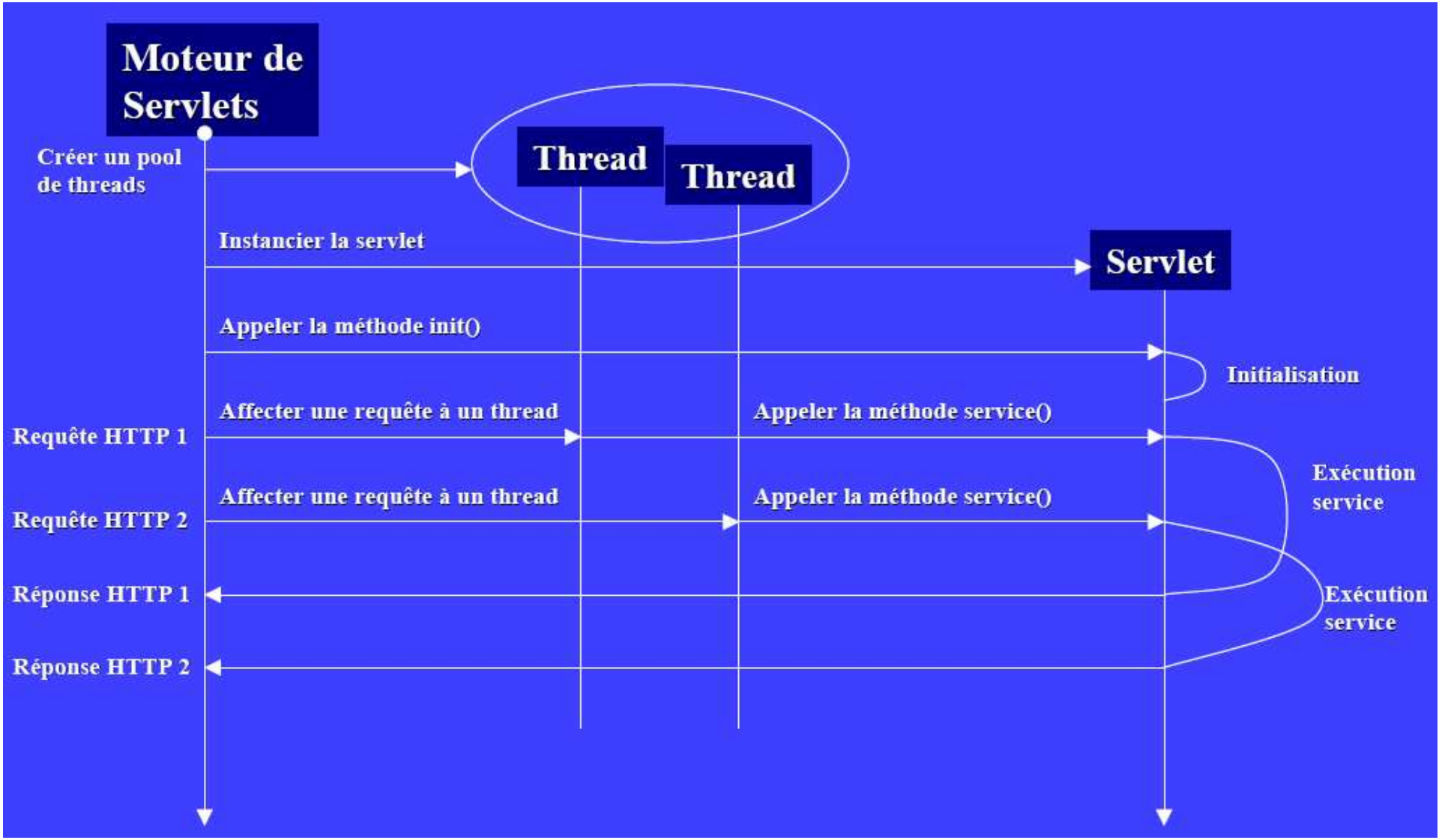
How the parameters are sent?

- It depends of the way (the HTTP method)
- Method GET: in the URL
- Method POST: in the body of the HTTP request

	Advantages	Drawbacks
GET	See the URL => more easily keep it to play again, send it by mail, …	The data sent may be truncated (by the client or the variables of the server)
POST	1°) No limitations for the size of data sent 2°) Don't see values in the URL => more confidentiality	Don't put the values in the URL => more difficult to play again and keep it

- Usually POST is used

Exécution des servlets



Les Servlets

servlet = ?

- Une servlet est un programme (plug-in) à ajouter à un serveur (quel qu'il soit).
- Ce cours a trait à la programmation Java coté serveur (Java EE)
- Pour l'instant les serveurs acceptant des servlets sont plutôt des serveurs Web.
- Contre-exemple : une servlet pour un serveur de mail qui détruit les mails contenant des virus.

application web = ?

- Les servlets sont une des techniques utilisées pour construire des applications web.
- "A web application is a dynamic extension of a web or application server. There are two types of web applications:
  - Presentation-oriented: A presentation-oriented web application generates interactive web pages containing various types of markup language (HTML, XML, and so on) and dynamic content in response to requests.
  - Service-oriented: A service-oriented web application implements the endpoint of a web service."
- [`http://java.sun.com/javaee/5/docs/tutorial/doc/bnadr.html`]

application web = ?

- Euh en français :
- "Une application web est une extension dynamique d'un serveur web ou applicatif. Il y a deux types d'applications web :
  - les application web orientées présentation qui génèrent des pages web (HTML, XML) dynamiquement
  - les applications web orientées service : ce sont les web services
- [`http://java.sun.com/javaee/5/docs/tutorial/doc/bnadr.html`]

Motivation et historique

- Nécessité d'avoir des pages HTML dynamiques i.e. pages créées lors de la requête (météo, cours de la bourse, vente aux enchères, etc.)
- Technologie des scripts CGI.
- Le serveur Web demande à lancer un programme par le protocole CGI
- Inconvénient : nécessite de créer un process (sauf technique propriétaire)

CGI : la technique

- Le serveur Web, lorsqu'une URL de script CGI est demandée, passe la main au programme adéquat qui exécute le script.
- Ce programme génère la partie dynamique en HTML.
- La page HTML est complétée par le serveur Web et retournée au client Web.

Servlets

- La technique des CGI en Java
- MAIS
  - Sans créer de processus
  - toute la puissance des API Java (accès aux divers domaines de l'informatique : BD, multimédia, réseau, objets distribués, composants, etc.)
  - indépendance de la plate-forme (OS et machine)

Comment ça marche ?

- Le serveur (Web) possède désormais un interpréteur Java (JVM)
- => il n'y a pas de processus créé lors de l'exécution de code Java
- Cf. les clients Web possèdent un interpréteur Java permettant de lancer des applets.
- D'où le nom de servlets.

Moteurs de servlets (et de JSP)

- Pour exécuter des servlets (resp. des JSP), il faut un moteur de servlets (resp. de JSP) dans le serveur Web.
- Ces moteurs sont des plug-in pour des serveurs Web existants
- Souvent des serveurs Web eux mêmes
- Un bon candidat plug-in : tomcat (`http://tomcat.apache.org/`)

Serveurs Web et servlets

- Il existe des serveurs Web qui traitent les servlets (et JSP) :
  - IBM WebSphere
  - BEA WebLogic Server 10
  - JBoss
- Voir à `java.sun.com/products/servlet`

Tomcat

- Développé par la communauté qui implémente les spécifications des servlets et JSP.
- Téléchargeable (en version d'utilisation élémentaire) gratuitement à
  `http://tomcat.apache.org/download-60.cgi`
- Plug-in de Apache, Microsoft IIS, …
- Est aussi un mini-serveur Web.
- "Apache Tomcat 6.x is the current focus of development. It implements the Servlet 2.5 and JSP 2.1 specifications."

Bidouilles Tomcat

- Il faut aussi :
  - positionner la variable `JAVA_HOME` au répertoire racine de la hiérarchie du SDK.
    Exemple :
    ```
    set JAVA_HOME=C:\Applications\jdk
    ```
- Documentation (d'installation et tutorial) de tomcat à

`http://tomcat.apache.org/tomcat-6.0-doc/index.html`

Programmation des servlets

- Utilise deux paquetages :
  - `javax.servlet` : paquetage générique
  - `javax.servlet.http` : paquetage pour serveurs Web
- Ces paquetages **ne** sont **pas** dans Java SE
- Sont des paquetages supplémentaires qui sont apportés avec l'installation de Tomcat.
- Ils sont aussi intégrés dans Java EE voir à `http://java.sun.com/javaee/index.jsp`
- Pour ce cours sur les servlets, il n'est pas nécessaire d'installer Java EE

Documentation et tutorial

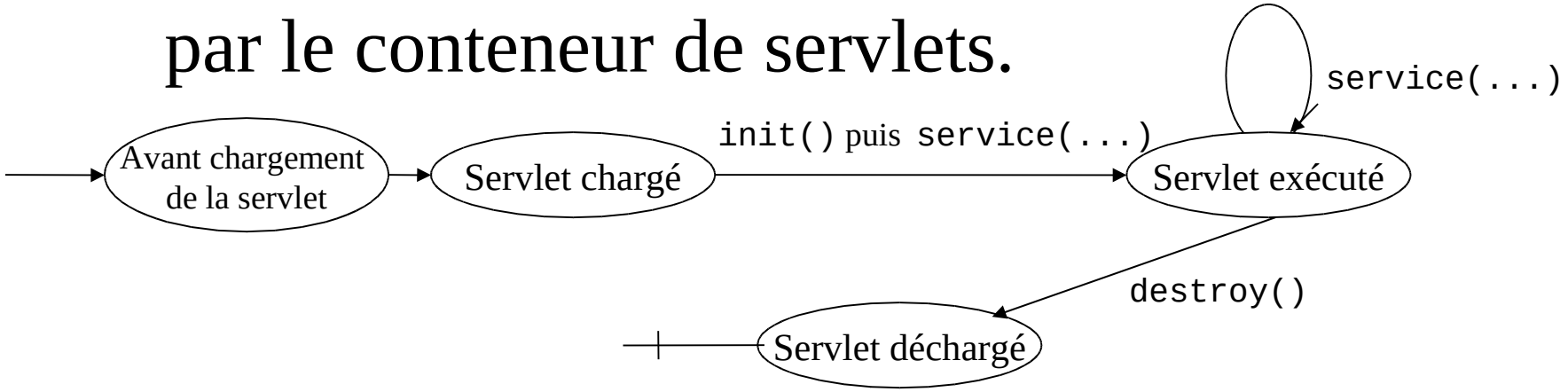
- Documentation à
  `http://java.sun.com/javaee/5/docs/api/index.html`
- Tutorial à :
  `http://java.sun.com/javaee/5/docs/tutorial/doc/bnafd.html`

Finalement une servlet =

- De manière similaire à une applet :
  - une servlet est une classe Java chargée par JVM intégrée au serveur.
  - Lorsque cette classe a été chargée, la JVM construit un objet de cette classe (instanciation).
  - Parfois (souvent) c'est cet objet qui est appelé une servlet.

Les états d'une servlet

- Cf. les états d'une applet. Le passage d'un état à un autre est automatique et est fait par le conteneur de servlets.



- Chargement = chargement en mémoire dans le moteur (conteneur) de servlets (i.e. la JVM).
- Le conteneur de servlets lance la méthode `service(…)` à chaque requête.

Méthodes fondamentales d'une servlet

- Les trois méthodes
  - `public void init()`,
  - `public void service(…)`,
  - `public void destroy()`

  sont définies dans la classe abstraite `javax.servlet.GenericServlet`.
- `service(...)` contient deux paramètres qui modélisent la requête et la réponse

Servlet pour serveur Web

- On a de nouveau les trois méthodes
  - `public void init()`,
  - `public void service(…)`,
  - `public void destroy()`
- spécialisées dans la classe abstraite `javax.servlet.http.HttpServlet` qui sont lancées automatiquement. Cette classe dérive de `javax.servlet.GenericServlet`

Construire une servlet pour serveur Web

- Cf. construction des applets
- On construit une classe qui hérite de la classe `javax.servlet.http.HttpServlet`
- On spécialise les méthodes qui nous intéressent
- On dépose cette classe « au bon endroit » du serveur Web

`service(...)` dans `HttpServlet`

- Contient deux paramètres :
```
protected void
service(HttpServletRequest req,
HttpServletResponse resp) throws
ServletException, java.io.IOException
```
- Le premier argument (`HttpServletRequest req`) modélise la requête
- Le second argument (`HttpServletResponse resp`) modélise la réponse

`service(...)` dans `HttpServlet` (suite)

- `service(...)` de `HttpServlet` redirige la requête suivant le type du message http.
- Méthode HTTP GET => `doGet(…)`
- Méthode HTTP POST => `doPost(…)`

`doGet(…)`, `doPost(…)`

- Ont les mêmes paramètres que `service(…)` de `HttpServlet` :

```
public void doXXX(HttpServletRequest
   request, HttpServletResponse
   response) throws ServletException,
java.io.IOException
```

- En général on spécialise l'une des deux méthodes et l’autre méthode appelle cette première.

Récupérer les données de la requête

- La requête envoie ses données par http
- Souvent des champs d'un formulaire
- On récupère ces données par `String getParameter(String` *`nomComposantFormulaire`*`)`
- sur `request`
- Cf. les applets

Récupérer les données de la requête (2/2)

- Plus précisément si on a, dans la page HTML :

```
Login : <input type="text" name="login" />
```

on doit avoir, dans les méthodes doXXX( ) de la servlet :

respecter la casse (majuscules et minuscules significatives !)

```
String nomLogin = request.getParameter("login");
```

Envoyer une réponse

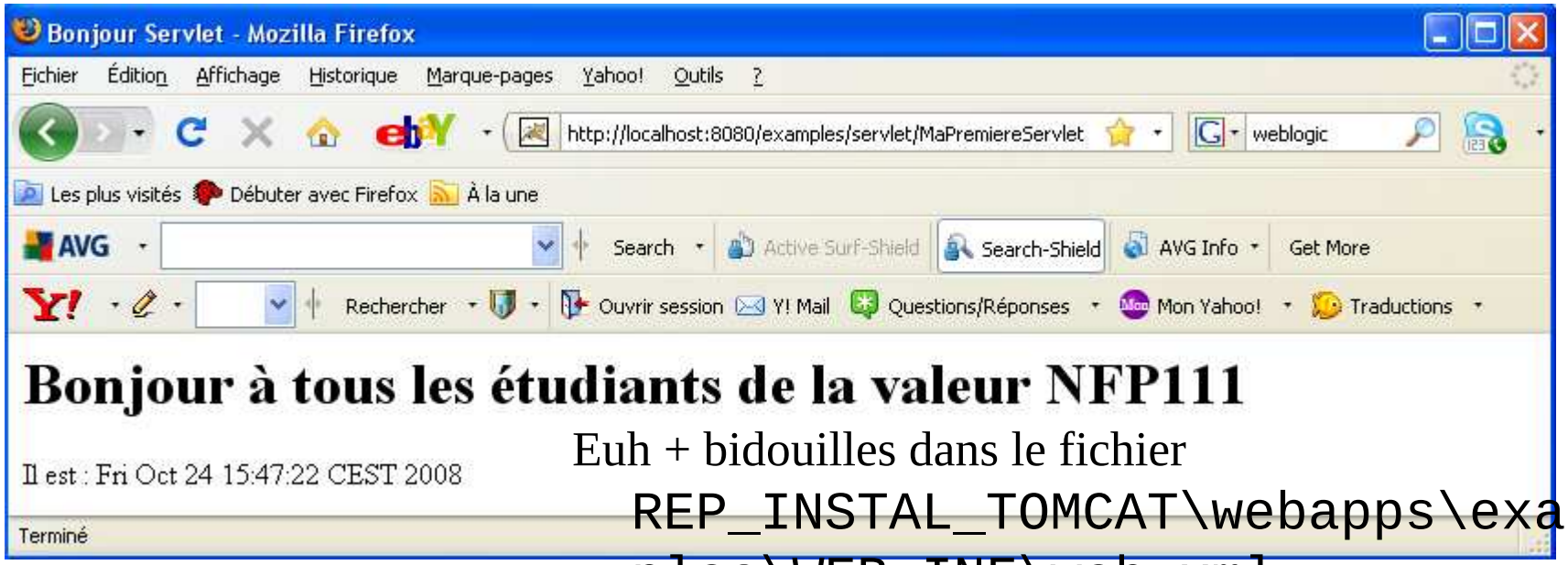
- On positionne le type MIME par : `void setContentType(String type)`
- On récupère le canal de sortie texte par : `PrintWriter getWriter()`
- sur `response`

Une servlet : code complet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MaPremiereServlet extends HttpServlet {

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

          // Etape 1.  Spécifier le type MIME du contenu de la réponse
          response.setContentType("text/html");

          // Etape 2.  Récupère le PrintWriter pour envoyer des données au client
          PrintWriter out = response.getWriter();

          // Step 3.  Envoyer l'information au client
          out.println("<html>");
          out.println("<head><title>Bonjour Servlet</title></head>");
          out.println("<body>");
          out.println("<h1>Bonjour aux étudiants de l'UE GLG203</h1>");
          out.println("Il est : " + new java.util.Date());
          out.println("</body></html>");
     }
     public void doPost(HttpServletRequest request, HttpServletResponse response)
       throws ServletException, IOException {
     doGet(request, response);
     }
}
```

Démonstration

- On lance le serveur Web
- La servlet compilée est rangée sous `REP_INSTAL_TOMCAT\webapps\examples\WEB-INF\classes`
- correspond à l'URL : `http://localhost:8080/examples/servlet/MaPremiereServlet`



Euh + bidouilles dans le fichier `REP_INSTAL_TOMCAT\webapps\examples\WEB-INF\web.xml`

Quelques précisions

- Pour compiler on peut utiliser le script (qu'il faut évidemment adapter !) :

```
set TOMCAT_HOME=C:\Applications\Tomcat
set OLD_CLASSPATH=%CLASSPATH%
set CLASSPATH=%TOMCAT_HOME%\lib\servlet-api.jar;%CLASSPATH%
javac MaPremiereServlet.java
copy MaPremiereServlet.class %TOMCAT_HOME%\webapps\examples\WEB-INF\classes
set CLASSPATH=%OLD_CLASSPATH%
```

Les bidouilles du `web.xml` de l'application web dans tomcat 6.x

- Le fichier `web.xml` du répertoire `WEB-INF` de l'application web configure l'application web.
- L'élément `<servlet>` associe un nom à une classe Java servlet
- L'élément `<servlet-mapping>` associe à ce nom l'URL relative qui permet d'atteindre cette servlet

```
<servlet>
     <servlet-name>nomQuelconque</servlet-name>
     <servlet-class>MaPremiereServlet</servlet-class>
</servlet>

<servlet-mapping>
     <servlet-name>nomQuelconque</servlet-name>
        <url-pattern>/servlet/MaPremiereServlet</url-pattern>
</servlet-mapping>
```

Architecture d'une application web 1/3

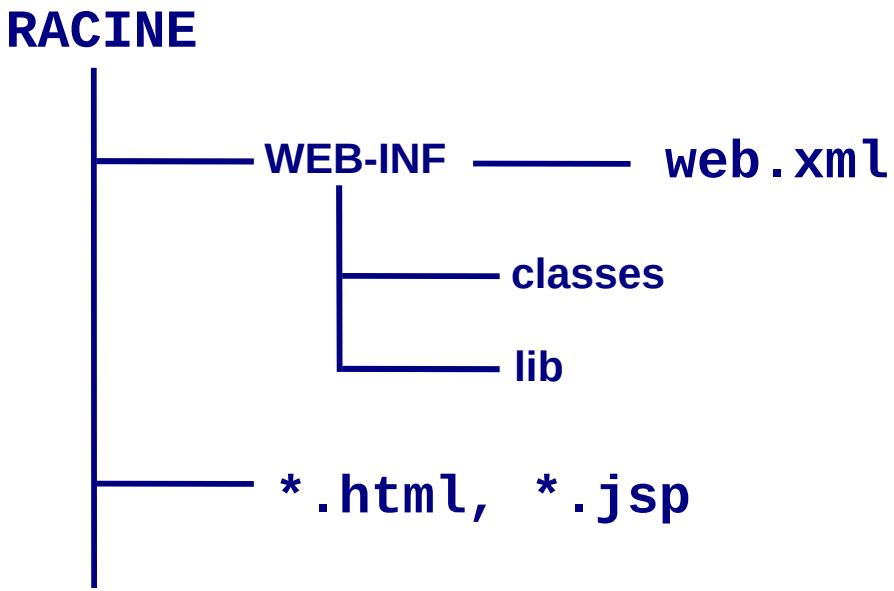
- Une bonne référence est : `http://tomcat.apache.org/tomcat-6.0-doc/appdev/deployment.html`
- Depuis la version 2.2 des spécifications des servlets, les ressources doivent être rangées comme indiqué ci dessous et diapos suivantes.
- Un site web est construit dans un arbre commençant au noeud `RACINE` (eh oui !) : c'est le "document root of your application".
- Ce noeud est associé à un "context path" pour les URL.
- Exemple : si le "context path" de votre application est `/catalog`, le fichier `index.html` du répertoire `RACINE` est accessible par une URL se terminant par `/catalog/index.html`

Architecture d'une application web 2/3

- Les fichiers `.html` et `.jsp` doivent être rangés à partir de `RACINE`.
- `/WEB-INF/web.xml` : le fichier descripteur de déploiement de votre application web
- `/WEB-INF/classes/` : le répertoire racine de rangement des `.class` (servlets compilés, etc.). Si les classes sont dans des packages, la hiérarchie des packages doit être respectée à partir de `/WEB-INF/classes/`. Par exemple la classe `cnam.ihm.MaServlet` doit être mise dans `/WEB-INF/classes/cnam/ihm/MaServlet.class`

Architecture d'une application web 3/3

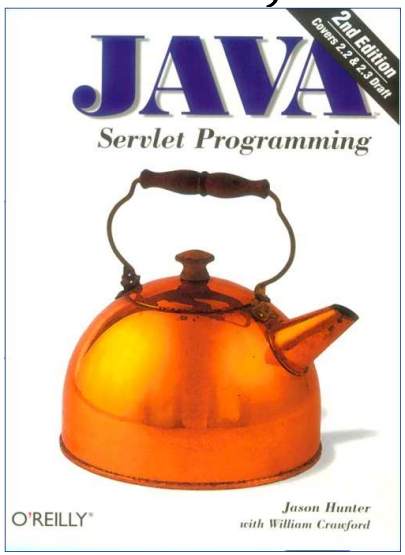
- `/WEB-INF/lib/` : le répertoire contenant les .jar nécessaires à votre application web (driver JDBC, etc.)



- On peut mettre tout cela dans un fichier compressé : un `.war`

Bibliographie

- Page de départ de la technologie servlets : `http://java.sun.com/products/servlet/index.html`
- Java servlets, Jason Hunter, ed O'Reilly traduit en français



Fin