Java EE

Cours de 2e année ingénieur
Spécialisation « Génie Informatique »

fabien.romeo@fromeo.fr
http://www.fromeo.fr

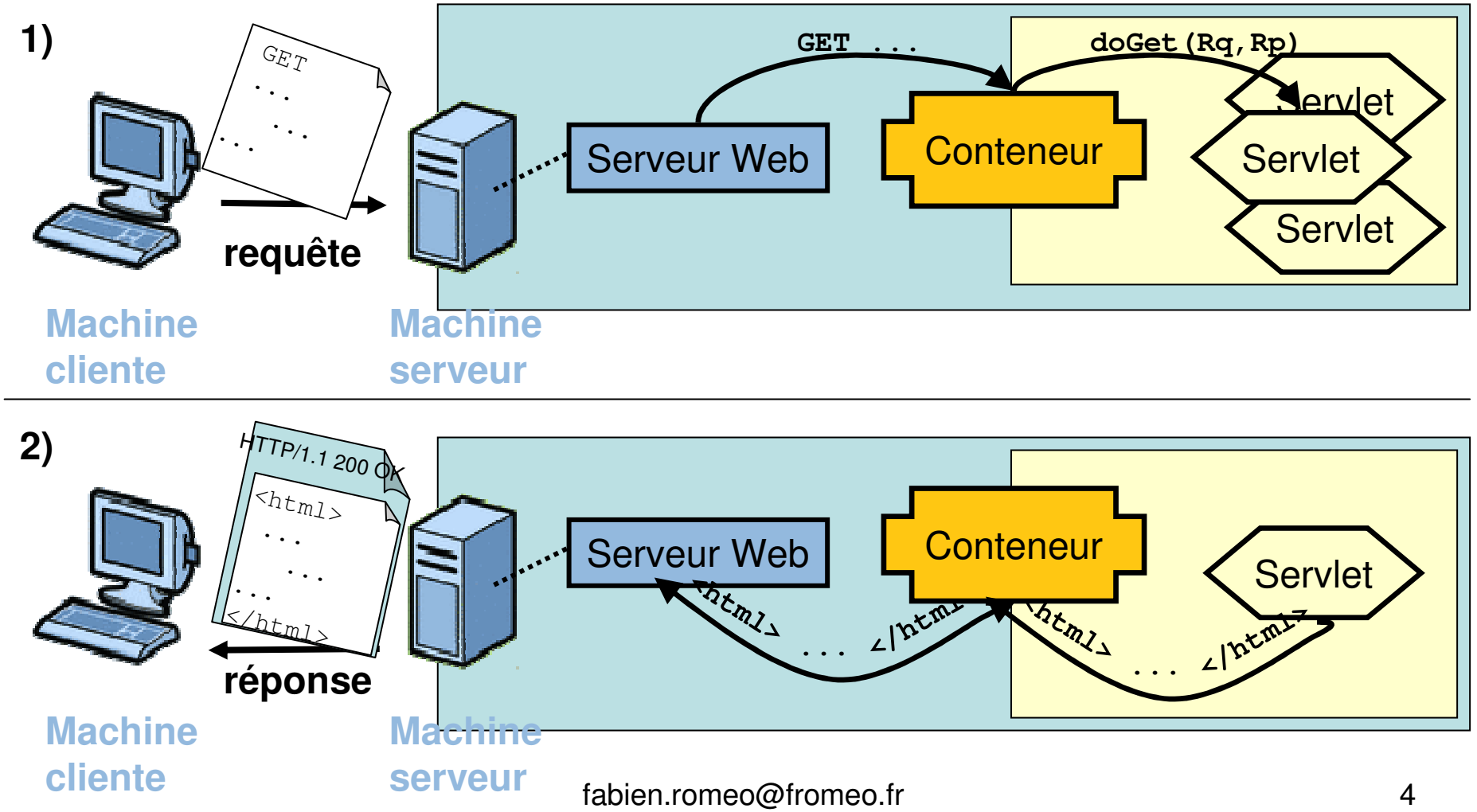
Introduction aux servlets

Conteneur

- La notion de conteneur se retrouve dans de nombreuses technologies
  - Servlet, Applet, MIDlet, Xlet, (**-let*), EJB, …
- Un conteneur est un composant logiciel *système* qui contrôle d'autres composants, dits *métier*
  - Tomcat est un exemple de conteneur
  - Les servlets n'ont pas de méthode **`main()`**, ils sont contrôlés par le conteneur Tomcat
  - Les requêtes ne sont pas adressées aux servlets mais au conteneur dans lequel ils sont *déployés*

Application Web avec un conteneur

- Le serveur Web a besoin d'aide pour faire du dynamique



Pourquoi un conteneur ?

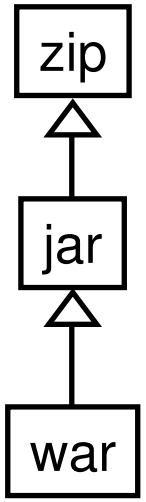
- Pour oublier le cours de « réseau » !
- Un conteneur fournit pour les Servlets
  - Un support pour la communication
    - Pas besoin de ServerSocket, Socket, Stream, …
  - La gestion du cycle de vie
  - Un support pour le Multithreading
    - Création automatique des Threads
  - Un support pour la sécurité
  - Un support pour les JSP

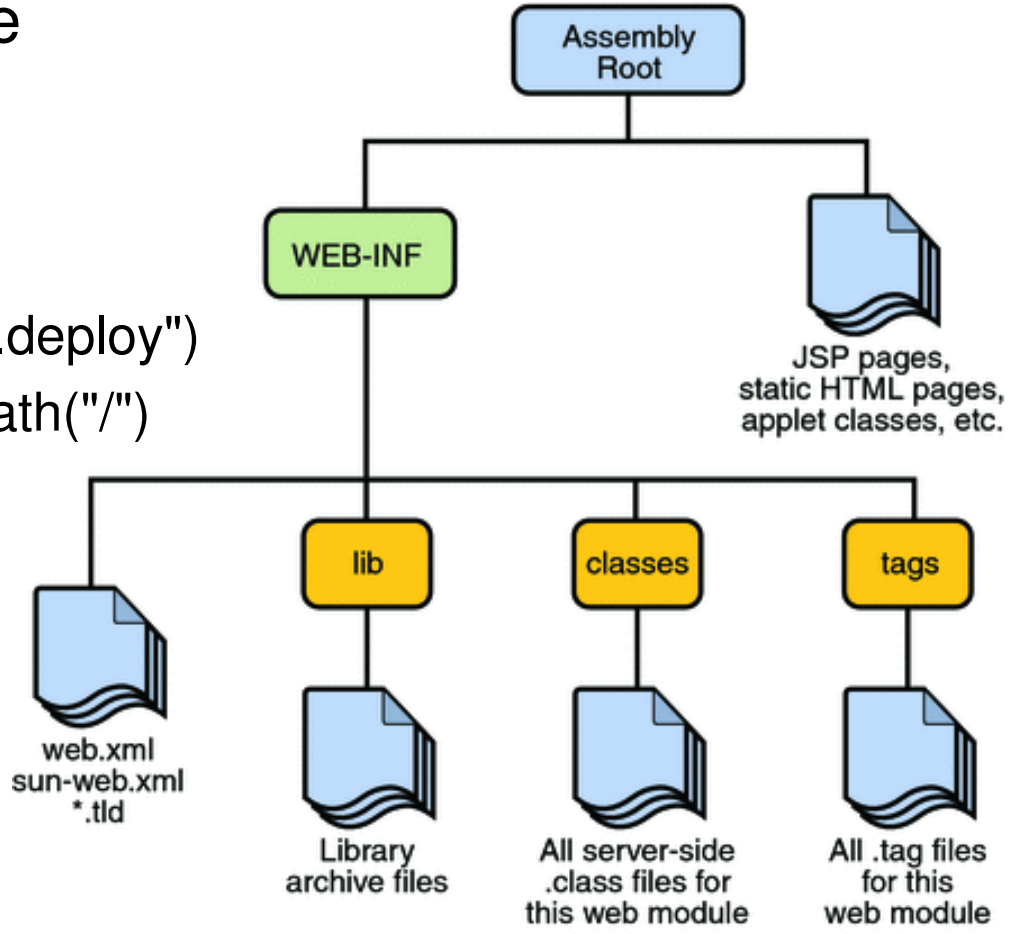
Notion de module Web

- Un servlet ne peut pas être déployé directement dans un conteneur, il doit faire partie d'un module Web
- Un module Web est un ensemble de librairies, de fichiers de configurations, de code Java (bytecode des servlets…), …
- Le module Web est l'unité de déploiement dans le conteneur

Structure d'un module Web (.war)

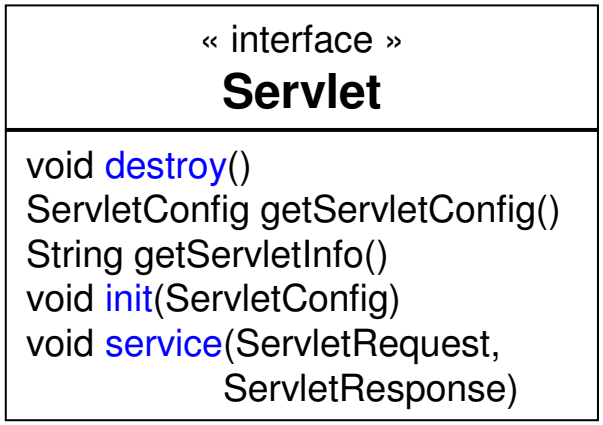
- Automatisé dans Eclipse
  - File / Export…
    Web / WAR file
- Eclipse-Tomcat plugin
  - System.getProperty("wtp.deploy")
  - ServletContext.getRealPath("/")





Servlets

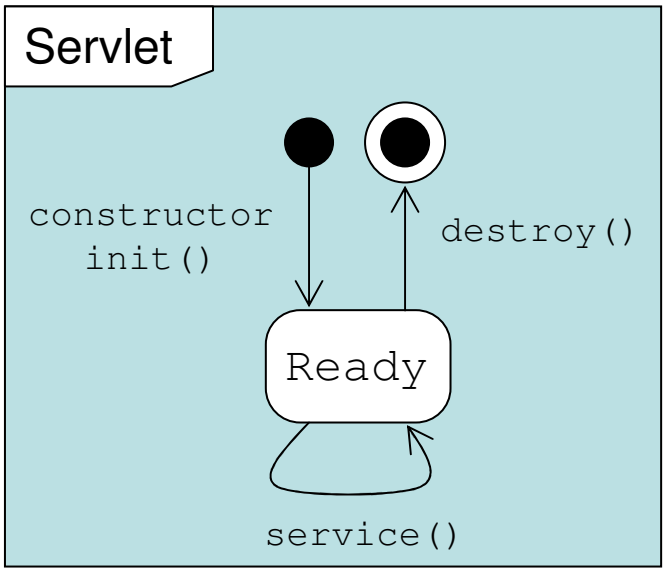
- Un servlet est un objet qui peut être manipulé par le conteneur via l'interface suivante :



- Lorsque le conteneur reçoit une requête, il la transmet au servlet qui correspond à l'URL pour que la requête soit traitée effectivement

Cycle de vie d'un servlet

1. Chargement de la classe
2. Instanciation du servlet
   - constructeur par défaut
3. Appel de init()
4. Appel(s) de service()
   - 1 thread par requête
5. Appel de destroy()



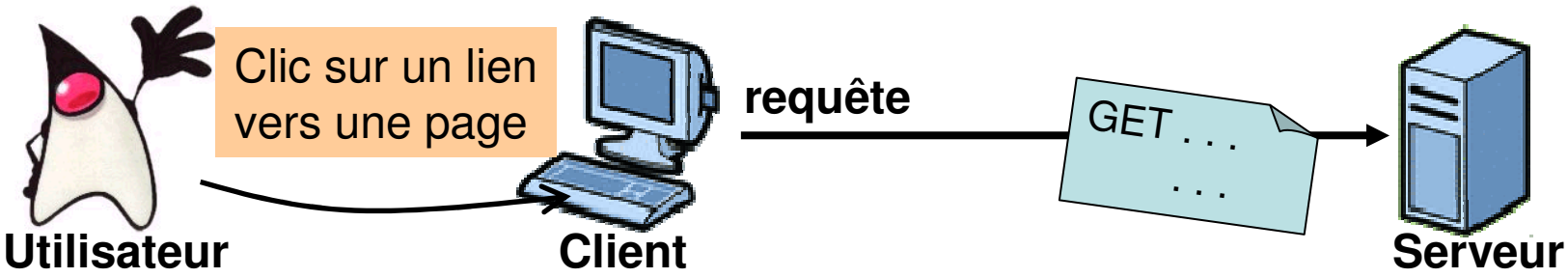
La méthode `service()`

- Lors de la réception d'une réquête, le conteneur crée
  - un objet `ServletRequest` (la requête), et
  - un objet `ServletResponse` (la réponse)
- Le conteneur appelle ensuite la méthode `service()` avec ces deux objets en paramètres pour permettre au servlet de répondre à la réquête du client

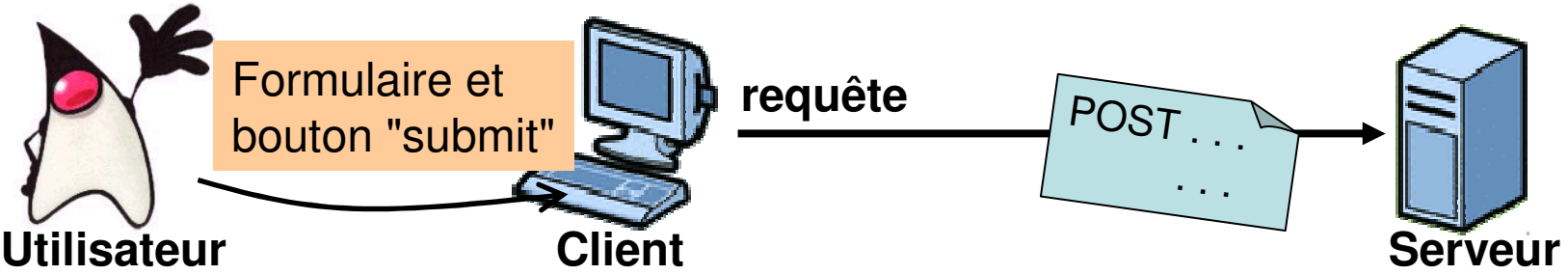
Requêtes HTTP (rappel)

- Deux méthodes principales : GET et POST

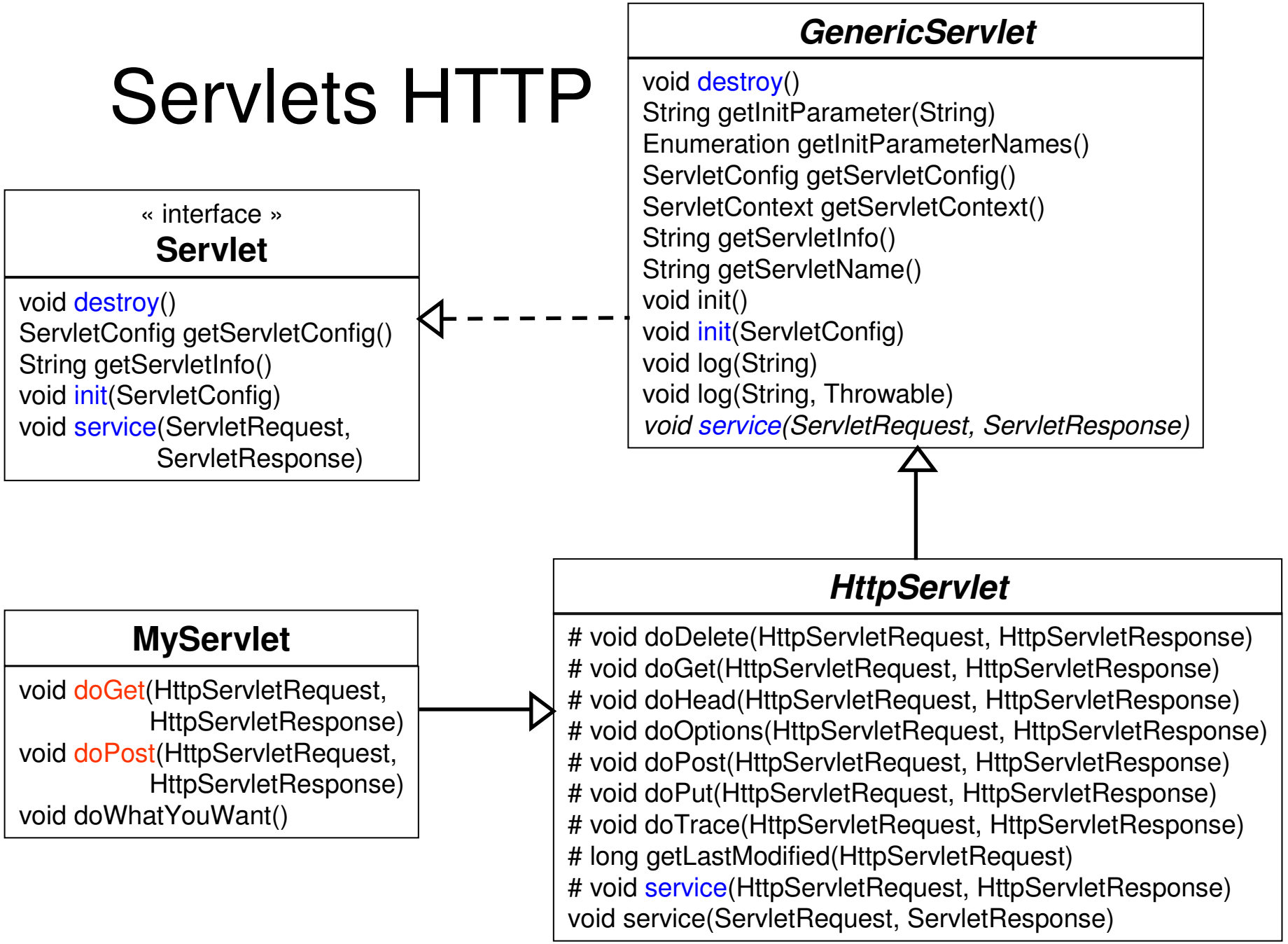
GET



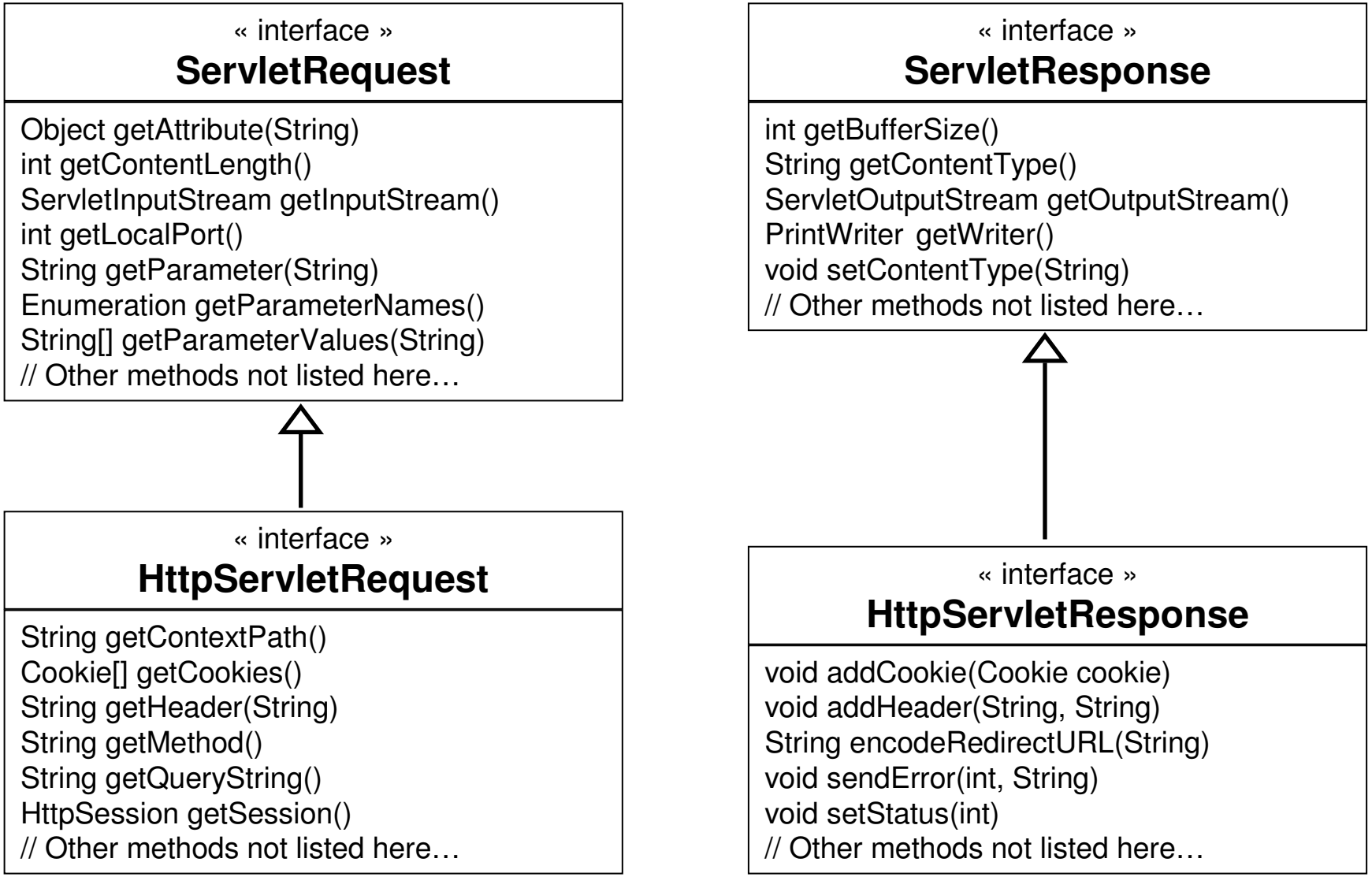
GET ou **POST**



Servlets HTTP



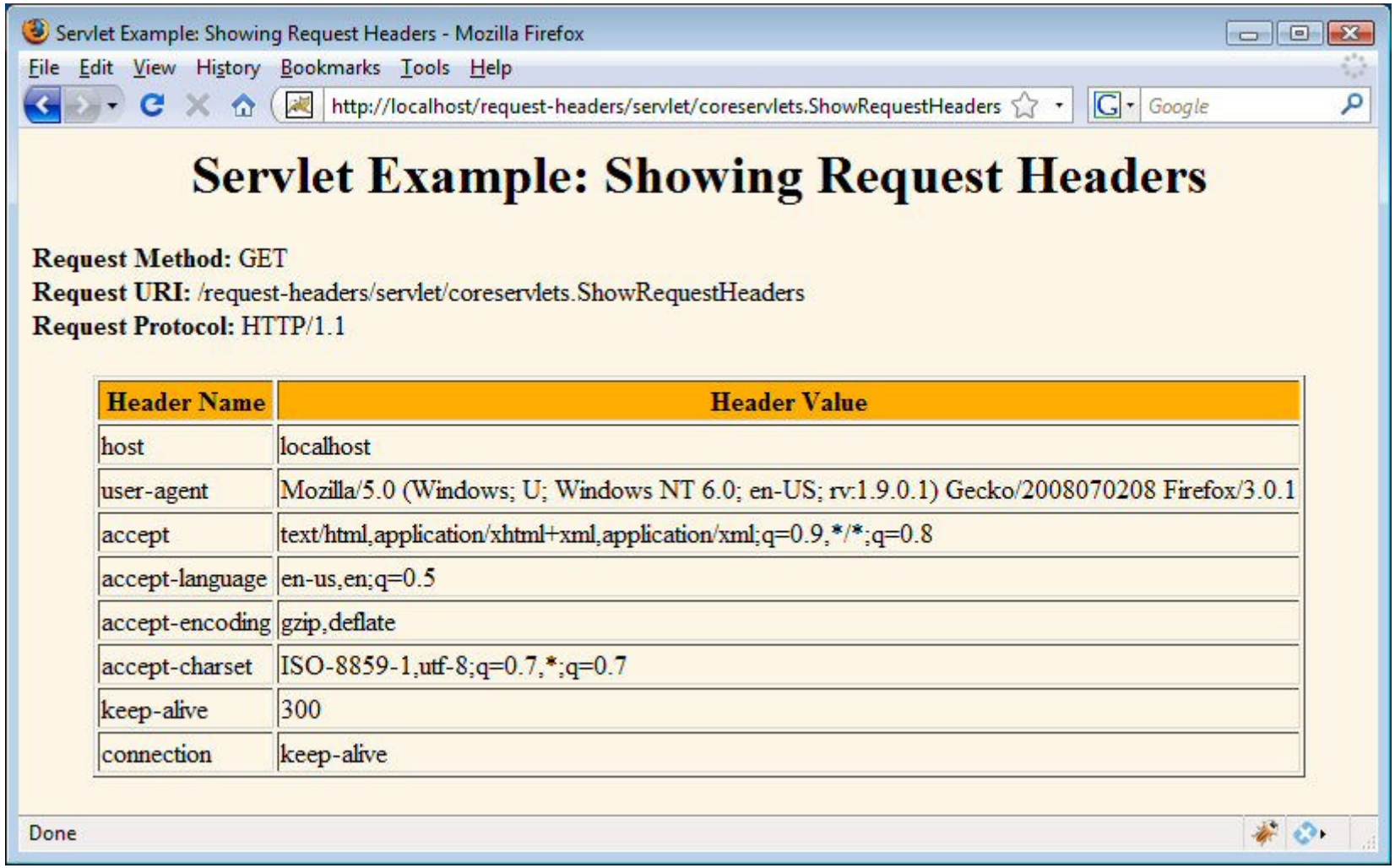
Requêtes et Réponses HTTP



Entêtes d'une requête GET

```
public class ShowRequestHeaders extends HttpServlet {
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws IOException {
    //...
    out.println(doctype +
        "<html>\n<head><title>" + title + "</title></head>\n" +
        "<body>\n<h1>" + title + "</h1>\n" +
        "<b>Request Method: </b>" + request.getMethod() + "<br />\n" +
        "<b>Request URI: </b>" + request.getRequestURI() + "<br />\n" +
        "<b>Request Protocol: </b>" + request.getProtocol() + "<br />\n" +
        "<table>\n" +
        "<tr><th>Header Name</th><th>Header Value</th></tr>");
    Enumeration<String> headerNames = request.getHeaderNames();
    while(headerNames.hasMoreElements()) {
        String headerName = headerNames.nextElement();
        out.println("<tr><td>" + headerName + "</td>");
        out.println("<td>" +
                    request.getHeader(headerName) +
                    "</td></tr>");
    }
    out.println("</table>\n</body></html>");
   }
}
```

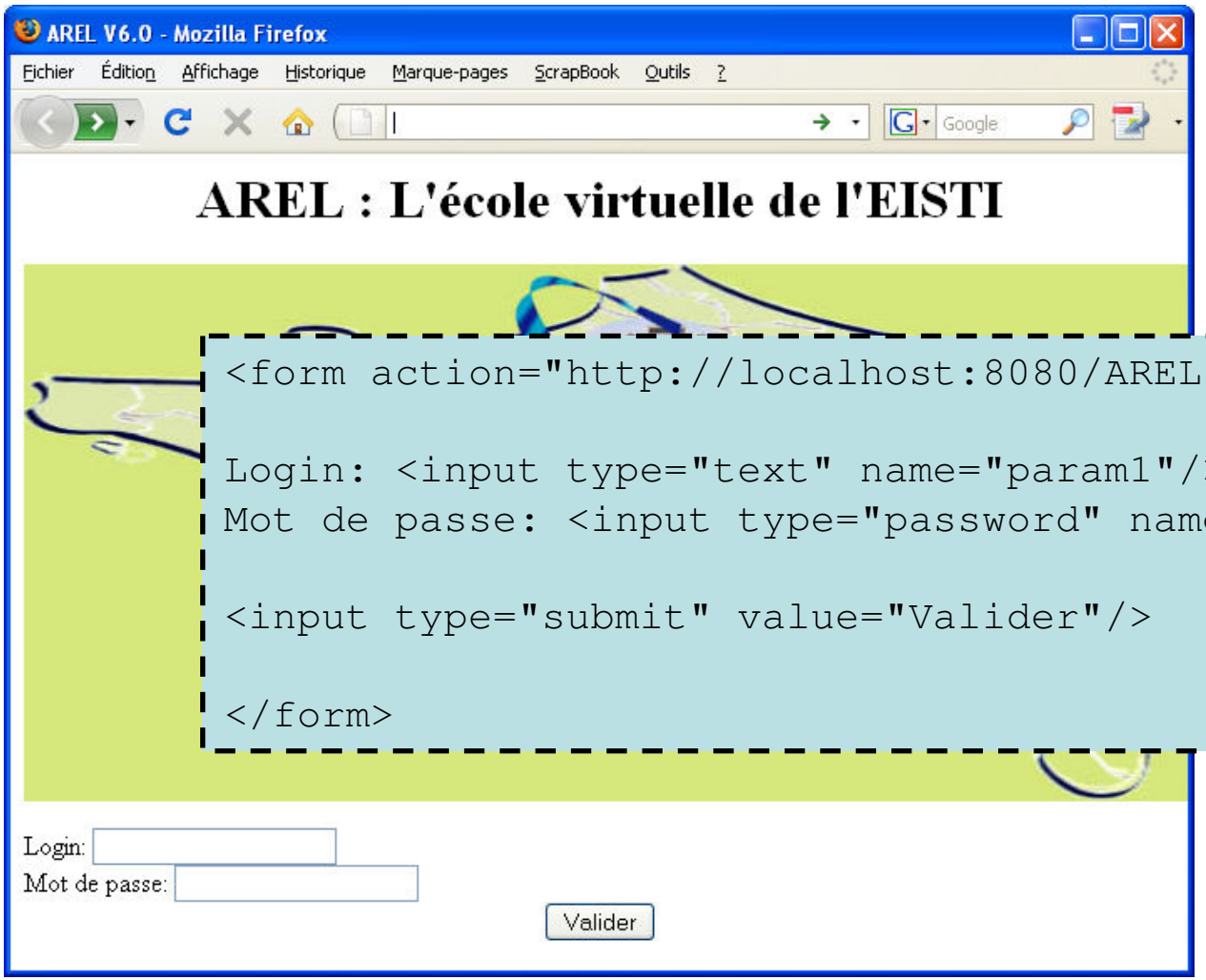
Entêtes d'une requête GET (2)



Entêtes d'une requête POST

```
public class ShowRequestHeaders extends HttpServlet {
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response) throws IOException {

        doGet(request,response);

    }
}
```

Formulaire GET



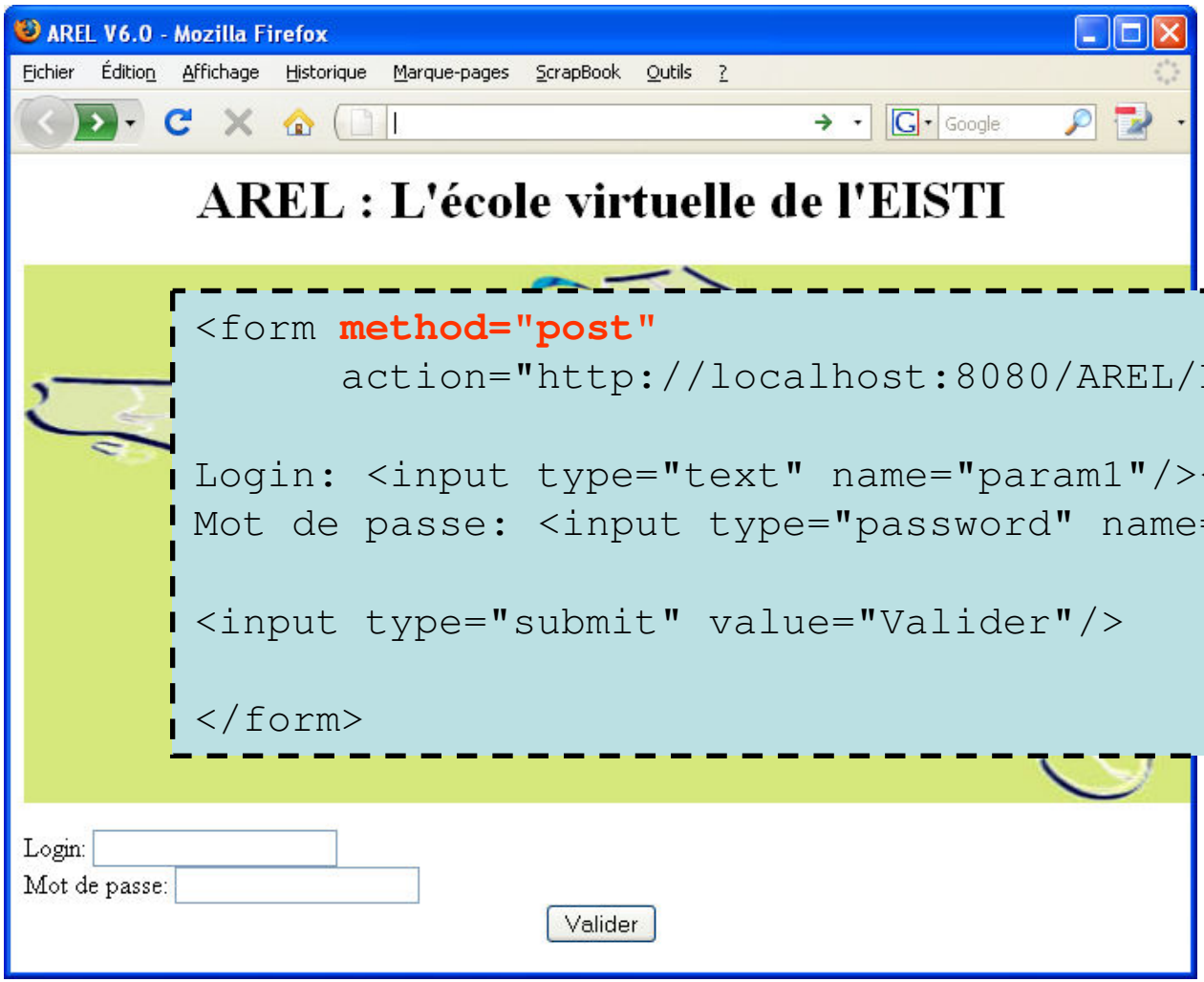

```
<form action="http://localhost:8080/AREL/LogServlet">

Login: <input type="text" name="param1"/><br/>
Mot de passe: <input type="password" name="param2"/><br/>

<input type="submit" value="Valider"/>

</form>
```

Traitement formulaire GET

GET /AREL/LogServlet?param1=monlogin&param2=monpass HTTP/1.1
host: localhost:8080
user-agent: Mozilla/5.0 (…) Gecko/2008092417 Firefox/3.0.3
accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
…

```
public class LogServlet extends HttpServlet {
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws IOException {
    String login = request.getParameter("param1") ;
    String password = request.getParameter("param2") ;
    if (checkUserAndPassword(login, password)) {
        grantAccessTo(login);
    } else {
        sendAuthentificationFailure(login);
    }
  }
}
```

Formulaire POST




```
<form method="post"
      action="http://localhost:8080/AREL/LogServlet">

Login: <input type="text" name="param1"/><br/>
Mot de passe: <input type="password" name="param2"/><br/>

<input type="submit" value="Valider"/>

</form>
```

Traitement formulaire POST

```
POST /AREL/LogServlet HTTP/1.1
host: localhost:8080
user-agent: Mozilla/5.0 (…) Gecko/2008092417 Firefox/3.0.3
accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
...
content-type: application/x-www-form-urlencoded
content-length: 30

param1=monlogin&param2=monpass
```

```
public class LogServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response) throws IOException {

        doGet(request,response);

    }
}
```

Paramètres de formulaires

```
public class ShowParameters extends HttpServlet {
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws IOException {
    //...
    out.println("<b>Query String: </b>" +
                request.getQueryString() + "<br />");

    out.println("<table>");
    Enumeration<?> parameterNames = request.getParameterNames();
    while(parameterNames.hasMoreElements()) {
        String parameterName = (String) parameterNames.nextElement();
        out.println("<tr><td>" + parameterName + "</td>");
        String[] paramValues = request.getParameterValues(parameterName);
        String paramValuesString = "";
        for(int i = 0; i < paramValues.length; i++) {
            paramValuesString += paramValues[i] + ";";
        }
        out.println("<td>" + paramValuesString + "</td>");
    }
    out.println("</table>");
  }
}
```

Vérification de formulaires

- Données manquantes
  - Champ manquant dans le formulaire
    - **`getParameter`** retourne null
  - Champ renvoyé vide
    - **`getParameter`** retourne une chaine vide (ou une chaine avec des espacements)

```
String param = request.getParameter("someName");
if ((param == null) || (param.trim().equals(""))) {
  doSomethingForMissingValues(...);
} else {
  doSomethingWithParameter(param);
}
```

- Données malformées
  - Chaine non vide mais dans le mauvais format (ex: code HTML si le résultat doit être affiché)

Upload de fichiers

- Formulaire HTML
  - `<input type="file" name="nameFile" />`
  - `<form method="post" enctype="multipart/form-data" action="/servlet">`
  - Le choix du *enctype* impacte les autres champs du formulaire
    - **`request.getParameter("name")` ne fonctionne plus**
- Côté Servlet
  - Bas-niveau : parser l'inputStream
    `request.getInputStream()`
  - Haut-niveau : utiliser une librairie
    ex: Commons FileUpload du projet Jakarta
    (http://commons.apache.org/fileupload/)

Commons FileUpload

- Développement
  - import org.apache.commons.fileupload.*
  - import org.apache.commons.fileupload.servlet.*
- Déploiement
  - Dans le module Web de l'application…
    - *ie.* même chose pour chaque application !
  - …dans le répertoire WEB-INF/lib
    - commons-fileupload-1.2.1.jar
    - commons-io-1.4.jar

Upload simple (1)

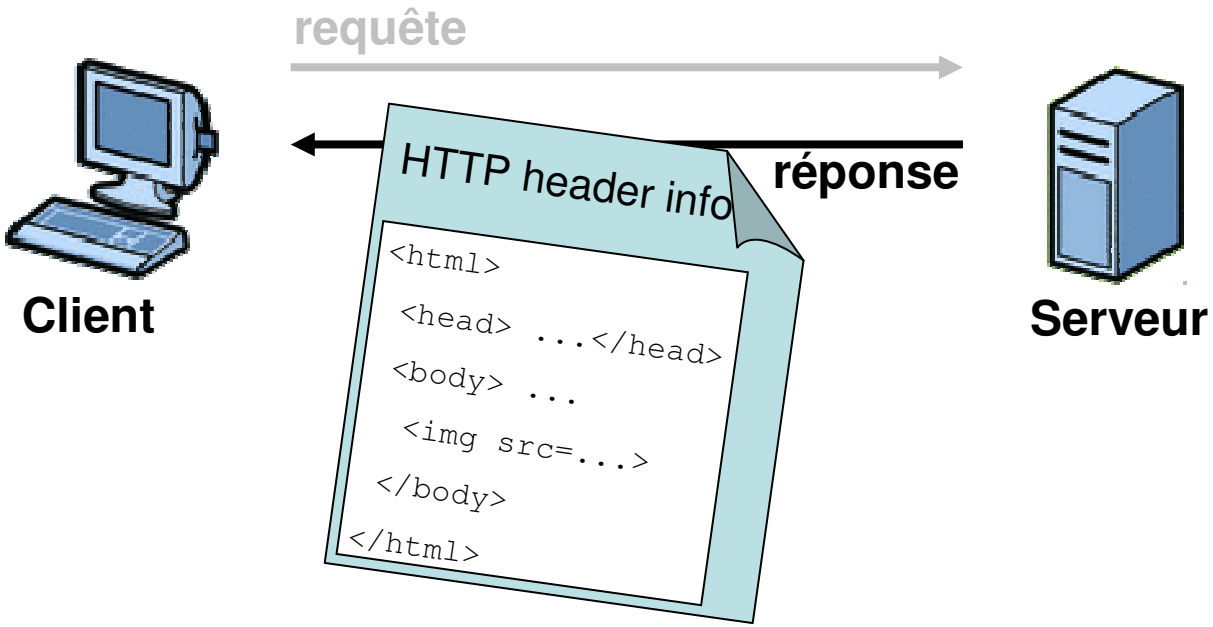
```
public void doPost(HttpServletRequest request,
                   HttpServletResponse response) throws IOException {
 //...
 // check file upload request
 if ( ServletFileUpload.isMultipartContent(request) ) {

     // create a factory for disk-based (large) file items
     FileItemFactory fileItemFactory = new DiskFileItemFactory() ;
     fileItemFactory.setSizeThreshold(40960); /* the unit is bytes */

     // create a new file upload handler
     ServletFileUpload servletFileUpload =
         new ServletFileUpload(fileItemFactory);
     servletFileUpload.setSizeMax(81920); /* the unit is bytes */

     // parse the request
     // ...              --->
 }
}
```

Upload simple (2)

```
// parse the request
try {
    List<?> fileItemsList = servletFileUpload.parseRequest(request);
    // Process file items
    Iterator<?> it = fileItemsList.iterator();
    while (it.hasNext()){
      DiskFileItem fileItem = (DiskFileItem)it.next();
      if (fileItem.isFormField()){ // classic form field (name=value)
        out.println("<b>Form field:</b><br />\n" +
                    fileItem.getString() + "<br/>");
      } else{ // uploaded file
        out.println("<b>File:</b><br />\n<pre>" +
                    fileItem.getString() + "</pre><br/>");
        // ex: save on disk
        File dest = new File(directoryPath,fileName);
        FileOutputStream fos = new FileOutputStream(dest);
        fos.write(fileItem.get());
        fos.close();
      }
    }
} catch (FileUploadException e) {e.printStackTrace();}
```

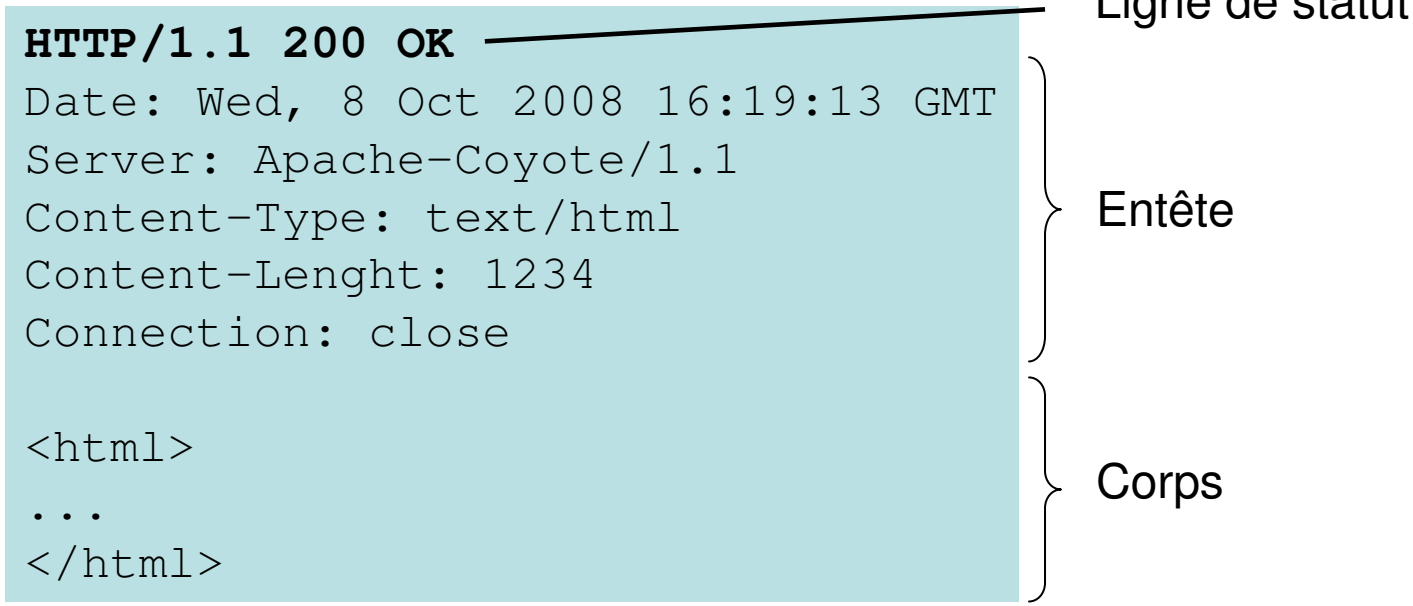
Réponse HTTP (rappel)

- Une réponse HTTP peut contenir du HTML
- HTTP rajoute des (meta)informations en entête du contenu de la réponse



Entête réponse HTTP

- Ex :



- Quelques codes réponses
  - 200 OK
  - 301 MOVED
  - 403 FORBIDDEN
  - 404 NOT FOUND
  - 503 SERVICE UNAVAILABLE

Status Codes

- response.setStatus(int statusCode)
  - Utiliser les constantes, pas d'entiers directement
  - Noms dérivés du message standard
    - Ex : SC_OK, SC_NOT_FOUND, etc
- response.sendError(int code, String msg)
  - Englobe le message dans un petit document HTML
- response.sendRedirect(String url)
  - Le code de status est alors 302
  - L'attribut « Location » est également généré dans l'entête de la réponse

Exemple sendError

```
public class LogServlet extends HttpServlet {
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws IOException {
    String login = request.getParameter("param1") ;
    String password = request.getParameter("param2") ;

    if ((param1 == null) || (param1.trim().equals(""))) {
        response.sendError(HttpServletResponse.SC_NOT_FOUND,
                           "Empty login");
        return;
    }

    if (checkUserAndPassword(login, password)) {
        grantAccessTo(login);
    } else {
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED,
                           "Access Denied to " + login);
    }
  }
}
```

Exemple sendRedirect

```
public class WrongDestination extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    String userAgent = request.getHeader("User-Agent");
    if ((userAgent != null) &&
        (userAgent.contains("MSIE")) {
      response.sendRedirect("http://home.netscape.com");
    } else {
      response.sendRedirect("http://www.microsoft.com");
    }
  }
}
```

Exemple sendRedirect (2)

- Même URL de départ pour les deux

setContentType

Type	Meaning
application/msword	Microsoft Word document
application/octet-stream	Unrecognized or binary data
application/pdf	Acrobat (.pdf) file
application/postscript	PostScript file
application/vnd.ms-excel	Excel spreadsheet
application/vnd.ms-powerpoint	Powerpoint presentation
application/x-gzip	Gzip archive
application/x-java-archive	JAR file
application/x-java-vm	Java bytecode (.class) file
application/zip	Zip archive
audio/basic	Sound file in .au or .snd format
audio/x-aiff	AIFF sound file
audio/x-wav	Microsoft Windows sound file
audio/midi	MIDI sound file
text/css	HTML cascading style sheet
text/html	HTML document
text/plain	Plain text
text/xml	XML document
image/gif	GIF image
image/jpeg	JPEG image
image/png	PNG image
image/tiff	TIFF image
video/mpeg	MPEG video clip
video/quicktime	QuickTime video clip

Générer un fichier Excel

```
public class ApplesAndOranges extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
      response.setContentType
        ("application/vnd.ms-excel");
      PrintWriter out = response.getWriter();
      out.println("\tQ1\tQ2\tQ3\tQ4\tTotal");
      out.println("Apples\t78\t87\t92\t29\t=SUM(B2:E2)");
      out.println("Oranges\t77\t86\t93\t30\t=SUM(B3:E3)");
  }
}
```