Les Servlets

Fred Hémery

IUT Béthune
Département
Réseaux &
Télécommunications

IC3 Application Web — 07/08

Plan

Plan

Les CGI

- CGI = Common Gateway Interface
- Permet à des programmes qui s'exécutent côté serveur de servir des requêtes ayant un contenu dynamique.
  - Interrogation d'une BD
  - Traitement en fonction de données saisies dans un formulaire
- Un programme CGI doit avoir les caractéristiques suivantes :
  - Capable de lire le flux de données d'entrée
  - Capable de traiter des chaînes de caractères
  - Capable d'écrire sur le flux de sortie
  - Exécutable ou interprétable par le serveur
  - Ex : shell, perl, tcl/tk
  - C, C++, Java

Les CGI

- Communication entre le navigateur et CGI
- Le programme CGI est responsable de l'en-tête http suivi d'une ligne vide
- Par exemple dans le cas

  D'une page html `Content-type : text/html`
  D'une image `Content-type : image/gif`
  D'un fichier quicktime `Content-type : text/quicktime`
- Exemple d'un script en shell

```
#!/bin/sh
echo "Content-type: text/html"
echo ""
echo "<html>"
echo "<head>"
echo "<title> Hello World </title>"
echo "</head>"
echo "<body>"
echo "hello world !"
echo "</body>"
echo "</html>"
```


Transmission des données

- La requête peut contenir des données qui ont été saisies par le client dans un formulaire (`<form> ... </form>`)
- Chaque élément du formulaire auquel on a donné un nom unique fournit une expression de la forme suivante :

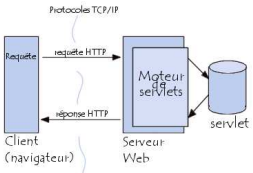
  `nom=valeur`
- Les données sont transmises au serveur dans l'URL lorsque la méthode `GET` est utilisée (par défaut). Dans ce cas chaque élément est séparé du suivant par le caractère '&'

  `http://nomServeur/cgi-bin/script.cgi?champ1=val1&champ2=val2`
- Les données sont transmises au serveur par le corps de la requête lorsque la méthode `POST` est utilisée. Cela permet de transmettre des données d'une taille > 255 cars.
- Le traitement des données du coté serveur est plus délicat lorsque l'on utilise la méthode `POST`. On utilise, en général, des fonctions spécifiques(`getParameter()`) pour les récupérées.

Plan

Présentation des servlets

- Les servlets sont l'équivalent côté serveur de l'applet côté client.
- Ce sont des programmes Java qui s'exécutent sur le serveur (comme les CGI) qui permettent de créer des réponses dynamiques.
- Les avantages d'une servlet par rapport au autres (CGI, ASP Active Server Page, ...)
  - Technologie java multi-plate-forme
  - Indépendance vis à vis du serveur : les servlets sont exécutées dans un conteneur de servlets.
  - Les servlets sont lancées, une seule fois, lors du lancement du conteneur ou lors du premier appel
  - Brique réutilisable



Le cycle de vie d'une servlet

- La vie d'une servlet est gérée par le conteneur.
- Le conteneur assure la gestion des requêtes clients, la transmission des requêtes à la servlet et l'envoi de la réponse vers le client. Pour cela la servlet doit implémenter des méthodes qui sont regroupées dans une interface.
- Interface d'une servlet : Les méthodes

`init()`	Initialise la servlet dans le conteneur qui la contrôle
`service()`	Traite les requêtes transmises parle conteneur
`getServletConfig()`	Renvoie une instance de la classe `ServletConfig` (informations dans la phase d'initialisation)
`getServletInfo()`	Renvoie une chaîne de caractères qui précise la fonction de la servlet
`destroy()`	Lors de la suppression de la servlet, cette fonction permet de récupérer les ressources allouées.

Le cycle de vie d'une servlet

- Le conteneur crée un pool de threads qui vont prendre en charge les requêtes ;
- La servlet est chargée au démarrage du conteneur ou lors de la 1ère requête
- La servlet est instanciée par le serveur et
  1. La méthode `init()` est invoquée par le conteneur
  2. La méthode `service()` est appelée à chaque requête. Les objets `Request` et `Response` lui sont passés en paramètre.
  3. Grâce à l'objet `Request`, la méthode `service()` va pouvoir analyser les informations en provenance du client.
  4. Grâce à l'objet `Response`, la méthode `service()` va fournir une réponse au client.
  5. La méthode `destroy()` est appelée lors du déchargement de la servlet.

Les paquetages de développement d'une servlet

- Deux paquetages sont nécessaires pour développer des servlets :
  - `javax.servlet`
  - `javax.servlet.http`
- La classe `javax.servlet.GenericServlet` est une classe abstraite qui implémente l'interface `javax.servlet.Servlet` (5 méthodes). La méthode `service()` n'a pas été implémentée.
- Pour écrire une servlet, il faut hériter directement ou indirectement de la classe `GenericServlet` et implémenter le code de la méthode `service()`.
- La classe `javax.servlet.http.HttpServlet` est une classe qui propose une solution pour le protocole `http`.
- La méthode `service()`, redirige le traitement de la requête vers une fonction (`doGet(...)`, `doPost(...)`, ...) qui correspond à la méthode du protocole http utilisée par le client (`GET`,`POST`,...)

Développer une servlet

- La solution la plus simple consiste à écrire une classe qui hérite de la classe `javax.servlet.http.HttpServlet` et de surcharger le code des méthodes (`doGet(...)`, `doPost(...)`,...) que l'on prendre en compte.

```
import javax.servlet.*;
import javax.servlet.http.*;
public class ServletDeBase extends HttpServlet {
      public void doGet(HttpServletRequest req,
        HttpServletResponse res) throws ServletException {
              // - lecture de la requête
              // - traitements
              // - envoi de la réponse
      }
}
```


Une première servlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*
public class PremiereServlet extends HttpServlet {
public void init() {
}
public void doGet(HttpServletRequest req,
                  HttpServletResponse res)
                        throws ServletException, IOException {
  res.setContentType("text/html");
  PrintWriter out = res.getWriter();
  out.println("<HTML>");
  out.println("<HEAD><TITLE> Titre </TITLE></HEAD>");
  out.println("<BODY>");
  out.println("Ma premi re servlet");
  out.println("</BODY>");
  out.println("</HTML>");
  out.close();
}
}
```


Quelques méthodes de la classe `HttpServletRequest`

`String getContextPath()`	renvoie la partie de l'URL qui fait référence au contexte de la requête.
`Cookie[] getCookies()`	renvoie le tableau des cookies transmis en même temps que la requête.
`String getHeader(String name)`	renvoie la valeur du mot clé de l'en-tête qui correspond au paramètre `name`.
`Enumeration getHeaderNames()`	renvoie une énumération des valeurs de chaque mot clé de l'en-tête.
`Enumeration getHeaders(String name)`	renvoie une énumération des valeurs du mot clé de l'en-tête qui correspond au paramètre `name`.
`String getMethod()`	renvoie la méthode utilisée pour transmettre la requête (`GET`, `POST`,...).

Quelques méthodes de la classe `HttpServletResponse`

`addCookie (Cookie cookie)`	ajoute un cookie à la réponse
`addHeader(String name, String url)`	ajoute un mot clé `name` avec la valeur `value`.
`String encodeURL(String url)`	ajoute à l'URL un champ qui correspond au numéro de session

Traitement des données d'un formulaire dans une servlet

- Un formulaire est codé de la manière suivante :

```
<form method="POST" action="userInfo">
  Nom : <input type="text" size="20" name="nom"><br>
  Prénom : <input type="text" size="20" name="prenom"><br>
  Age : <input type="text" size="2" name="age"><br>
  <input type="submit" value="Envoyer">
</form>
```

- Chaque élément a un nom unique qui se voit attribuer une valeur :

```
champ1=valeur1&champ2=valeur2&champ3=valeur3&champ4=valeur4
```

- Si le client utilise la méthode `GET`, les données seront transmises après l'URI

```
http://serveur/monAppli/userInfo?nom=valeur1&prenom=valeur2&age=valeu
```

- Si le client utilise la méthode `POST`, les données sont transmises dans le corps de la requête

Traitement des données d'un formulaire dans une servlet

- La servlet dispose de méthodes qui permettent de traiter facilement les paramètres
  - La méthode `getParameter():String getParameter(String name)` qui renvoie la valeur du nom du paramètre donné en argument ou une référence nulle si le nom du paramètre n'existe pas dans la requête.
  - La méthode `getParameterValues():String[] getParameterValues(String name)` qui renvoie le tableau des valeurs du nom du paramètre donné en argument ou une référence nulle sinon.
  - La méthode `getParameterNames():Enumeration getParameterNames()` qui renvoie une énumération des noms de paramètres transmis dans la requête.

Traitement des données d'un formulaire dans une servlet

- Un exemple

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

public class UserInfo extends HttpServlet

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

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HTML><BODY>" +
    "<H1>Recapitulatif des informations</H1>" +
    "<UL>" + "  <LI>Nom : " + request.getParameter("nom") + "" +
    "  <LI>Prenom : "+ request.getParameter("prenom") + "" +
    "  <LI>Age : "+ request.getParameter("age") + "" +
    "</UL>" + "</BODY></HTML>");
```


Réponse autre que du texte/HTML

- Un exemple

```
...
public class ServletDemo2 extends HttpServlet
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                  throws ServletException, IOException
    OutputStream out = response.getOutputStream();
    String filename = getServletContext().
         getRealPath("/") + "/images/defenseFumer.png";
    try
    byte[] buffer = new byte[1024];
    FileInputStream in = new FileInputStream(filename);
    int nbBytes = 0;
    while ((nbBytes = in.read(buffer)) != -1)
      out.write(buffer,0,nbBytes);
      response.setContentType("image/png");
     catch (Exception e)
      System.err.println(e.toString());
     finally
      out.close();
```


Les sessions

- Le protocole http ne maintient pas la connexion entre plusieurs requêtes (http<=1.0)
- Il faut mettre en place un suivi de session si l'on veut garder des informations partagées par plusieurs requêtes d'un client
- Plusieurs solutions existent :
  - La réécriture des URL
  - Création de champs formulaires cachés (hidden)
  - En utilisant les cookies
  - Les outils de suivi de sessions associés aux servlets
- Une session se définit comme un ensemble d'interactions apparentées entre un client unique et un serveur web
  - Achat en ligne
  - Accès à un compte bancaire

Les sessions

- La réécriture d'URL
- Dans la requête du client, des informations sur l'identification de la session sont ajoutées, par exemple :

```
...
<UL>
<li><a href="http ://www.serveur.fr/servlet/usrmenu?uid=toto>
      User Prefs</a> </li>
<li><a href="http ://www.serveur.fr/servlet/tsEntry?uid=toto>
      Time Sheets</a> </li>
</UL>
```


Les sessions

- Les champs de formulaires cachés
  - Les champs cachés d'un formulaire ne sont pas visibles et ne sont pas modifiables par l'utilisateur

    `<input type="hidden" name="uid" value="toto">`
  - Cette valeur sera récupérée par la servlet lors du traitement de la requête
- Les cookies
  - Les cookies sont de petits fichiers texte contenant des couples `clé=valeur`
  - Les cookies sont transmis par le serveur en même temps que la réponse dans l'en-tête http.
  - Le navigateur qui les reçoit, les stocke sur le disque local et les retransmet au serveur qui les avait transmis à chaque nouvelle requête et cela pendant la durée de vie du cookie.

Les sessions

- Le suivi de session en utilisant la classe `HttpSession`
  - La classe `HttpSession` permet de stocker des données d'une session dans le contexte d'une servlet. Cela évitera de transmettre ces informations à chaque fois au client. Il faut néanmoins associer la session à la requête en utilisant un identificateur. Pour cela on pourra utiliser un cookie ou alors la réécriture de l'URL.
  - Création d'une session : `getSession(true)` de la classe `HttpServletRequest`
  - Pour obtenir des informations d'une session : `getAttribute("cle")` de la classe `HttpSession`
  - Pour ajouter des informations dans une session : `setAttribute("cle", "valeur")` de la classe `HttpSession`

Plan

Introduction

- Les sessions attachées à une servlet permettent de gérer les requêtes entre le client et la servlet.
- La classe HttpSession peut être vue comme une table qui fait correspondre un objet en fonction d'une clé
- Comment faire si plusieurs servlets veulent partager des données communes ?
  - Utilisation d'une zone de données persistante (SGBD)
  - Utilisation de la notion de contexte d'une servlet
- Les servlets d'une application partagent le même contexte.

Configuration du contexte

- Le travail de configuration d'une servlet est assuré par le conteneur de servlets (par exemple `tomcat`).
- Le conteneur va placer les servlets d'une application web dans le même contexte.
- soit l'URL suivante :

```
http ://iut-gtr2 :8080/moteurRecherche/question ?mot=éléphant
```


Configuration du contexte

- Le travail de configuration d'une servlet est assuré par le conteneur de servlets (par exemple `tomcat`).
- Le conteneur va placer les servlets d'une application web dans le même contexte.
- soit l'URL suivante :

```
http ://iut-gtr2 :8080/moteurRecherche/question ?mot=éléphant
```

- `moteurRecherche` correspond au contexte (nom de l'application web)

Configuration du contexte

- Le travail de configuration d'une servlet est assuré par le conteneur de servlets (par exemple `tomcat`).
- Le conteneur va placer les servlets d'une application web dans le même contexte.
- soit l'URL suivante :

```
http ://iut-gtr2 :8080/moteurRecherche/question ?mot=éléphant
```

- `moteurRecherche` correspond au contexte (nom de l'application web)
- `question` nom de la servlet dans le contexte

Configuration du contexte

- Le travail de configuration d'une servlet est assuré par le conteneur de servlets (par exemple `tomcat`).
- Le conteneur va placer les servlets d'une application web dans le même contexte.
- soit l'URL suivante :

```
http ://iut-gtr2 :8080/moteurRecherche/question ?mot=éléphant
```

- `moteurRecherche` correspond au contexte (nom de l'application web)
- `question` nom de la servlet dans le contexte
- `mot=éléphant` paramètre de la servlet dans le contexte

Correspondance

- Correspondance entre son nom dans le contexte et la classe java qui l'implémente
  - `question` $\Longrightarrow$ `iutgtr.webAppli.moteurRecherche.Question.class`
- Correspondance entre nom de la servlet dans le contexte et comment elle est appelée dans une URL
  - `question` $\Longrightarrow$ `/question`
- Cela permet d'implémenter de manière évolutive le traitement d'une servlet sans modifier la configuration

Le contexte : traitement d'une requête

- Différentes situations :
  - Une requête en cours de traitement par une servlet peut être redirigée vers une autre servlet (forward)
  - Une requête en cours de traitement par faire intervenir d'autres servlets (input)
- la classe `ServletContext` dispose d'une méthode `getRequestDispatcher()` qui renvoie une instance de la classe `RequestDispatcher` et qui permet de rediriger (dispatcher) le traitement de la requête.
- la classe `RequestDispatcher` dispose (entre autres) de deux méthodes :
  - `forward` qui redirige le traitement d'une requête vers une autre servlet
  - `input` qui inclut le résultat du traitement d'une servlet dans la réponse qui sera envoyée au client.

Accès au contexte

- Les données partagées par l'ensemble des servlets d'un contexte sont accessibles par la classe `ServletContext`, dont une instance est obtenue à partir de la méthode `getServletContext()`.

```
ServletContext context = getServletContext()
```

- Le contexte gère une table qui fait correspondre à une clé $\Longrightarrow$ un objet (identique dans son principe à la `HttpSession`)

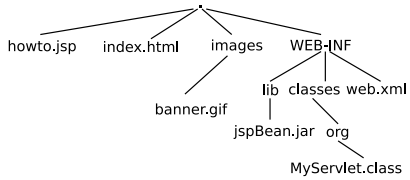
```
context.getAttribute(String nom) // renvoie un objet
```

- `GetAttributeNames()` renvoie une énumération des clés contenues dans le contexte.
- Le contexte peut être initialisé au travers du conteneur

```
context.getInitParameter(java.lang.String name) // renvoie un objet
context.GetInitParameterNames() // renvoie une énumération
                                // des paramètres d'initialisation
```

Développement d'une Application web

- L'arborescence des fichiers qui constituent votre application web dans la phase de développement doit suivre la structure suivante :
  - La racine de l'application (`~/appliWeb/jeu`)
  - `src` qui contient l'ensemble des fichiers sources
  - `webContent` qui contient les documents statiques `html`, `jsp`, `images`
  - `webContent/WEB-INF` qui contient le fichier de déploiement `web.xml`
- L'arborescence des fichiers qui constituent votre application web dans la phase d'exploitation :



Développement d'une Application web

- Le déploiement de l'application correspond à l'installation de l'application web que vous avez développée dans l'environnement d'un conteneur de servlet.
- Les consignes de déploiement sont stockées dans le fichier `web.xml`
- Il doit contenir au minimum :
  - La correspondance entre nom de classe contenant la servlet et son nom dans l'application web (balises servlet)
  - La correspondance entre le nom de la servlet dans l'application et le schéma URL qui va l'appeler (mapping)
- On trouve également
  - Des paramètres d'initialisations (nom de drivers SGBD)
  - Des modes de lancement de la servlet

`web.xml` : La déclaration des servlets

- Une servlet doit être déclarée dans l'application web

```
<servlet>
  <servlet-name>admin</servlet-name>
  <servlet-class>com.mycorp.AdminServlet
    </servlet-class>
  <load-on-startup>3</load-on-startup>
</servlet>
<servlet>
  <servlet-name>prems</servlet-name>
  <servlet-class>com.mycorp.PremiereServlet
</servlet-class>
</servlet>
<servlet>
  <servlet-name>catalog</servlet-name>
  <servlet-class>com.mycorp.CatalogServlet
    </servlet-class>
  <init-param>
    <param-name>catalog</param-name>
    <param-value>Spring</param-value>
  </init-param>
</servlet>
```


`web.xml` : Les associations des servlets

```
<servlet-mapping>
  <servlet-name>prems</servlet-name>
  <url-pattern>/premiere</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>catalog</servlet-name>
  <url-pattern>/catalog/*</url-pattern>
</servlet-mapping>
```


`web.xml` : Les associations des servlets

modèle de chemin	servlet
`/foo/bar/*`	`Servlet1`
`/baz/*`	`Servlet2`
`/catalog`	`Servlet3`
`*.bop`	`Servlet4`

chemin indiqué	servlet
`/foo/bar/index.html`	`Servlet1`
`/foo/bar/index.bop`	`Servlet1`
`/baz`	`Servlet2`
`/baz/index.html`	`Servlet2`
`/catalog`	`Servlet3`
`/catalog/index.html`	`"default" servlet`
`/catalog/racecar.bop`	`servlet4`
`/index.bop`	`servlet4`

`web.xml` : Les filtres

- Les filtres d'authentification
- les filtres de trace et de logging
- les filtres de conversion d'image
- les filtres de compression de données
- les filtres de cryptage
- les filtres d'analyse de contenu
- les filtres d'accès à certaines ressources
- les filtres de manipulation XML et XSL/T
- les filtres cache
- les filtres d'adaptation en fonction du type de données

`web.xml` : Les filtres

- La déclaration d'un filtre

```
<filter>
  <filter-name>Image Filter</filter-name>
  <filter-class>com.acme.ImageServlet</filter-class
</filter>
```

- L'association d'un filtre

```
<filter-mapping>
  <filter-name>Image Filter</filter-name>
  <servlet-name>ImageServlet</servlet-name>
</filter-mapping>

<filter-mapping>
  <filter-name>Logging Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```


`web.xml`

- Les fichiers de bienvenue

```
<welcome-file-list>
<welcome-file>index.html</welcome-file>
<welcome-file>default.jsp</welcome-file>
</welcome-file-list>
```

- les pages d'erreurs
- la déclaration du codage des caractères

web.xml : La sécurité

- L'expréssion d'une politique de sécurité déclarative peut être indiquer dans le fichier `web.xml`
- Elle est basée sur une authentification des utilisateurs
  - HTTP Basic Authentication
  - HTTP Digest Authentication
  - HTTPS Client Authentication
  - Form Based Authentication
- pour mettre en oeuvre
  - Le contrôle d'accès aux ressources
  - Le contrôle de l'intégrité des données
  - La confidentialité

web.xml : La sécurité

- Une contrainte de sécurité s'exprime en indiquant les éléments suivants :
  - la ressource (collection de ressources)
  - la méthode d'accès (POST, GET, PUT, DELETE, ...)
  - le rôle de l'utilisateur authentifié
  - le niveau de protection des données (intégrité, confidentialité)
- Le rôle indiqué dans la contrainte peut être associé avec un rôle attaché au serveur d'application

```
<security-role-ref>
<role-name>CONTRACTOR</role-name>
<role-link>manager</role-link>
</security-role-ref>
```


web.xml : La sécurité

```
<security-constraint>
<web-resource-collection>
<web-resource-name>wholesale</web-resource-name>
<url-pattern>/acme/wholesale/*</url-pattern>
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>
<auth-constraint>
<role-name>CONTRACTOR</role-name>
</auth-constraint>
<user-data-constraint>
<transport-guarantee>CONFIDENTIAL</transport-guarantee
</user-data-constraint>
</security-constraint>
```


Plan

Un exemple simple