Introduction à JHTML (Java Page Compilation 0.9)

Code: java-jhtml

Originaux

url: **http://tecfa.unige.ch/guides/tie/html/java-jhtml/java-jhtml.html**

url: **http://tecfa.unige.ch/guides/tie/pdf/files/java-jhtml.pdf**

Prérequis

- Java de base

  Module technique précédent: java-intro
- Avoir une idée du standard "CGI"

  Module technique précédent: cgi-intro
- Pages actives avec PHP (ou équivalent), conseillé mais pas obligatoire.

  Module technique précédent: php-html

Autres modules

Module technique suppl.: act-java-jsp

Module d'exercices: act-java-jhtml

Module technique suivant: java-servl

Objectifs

- Note: JHTML (alias JSP 0.9) est démodé !!
- Faire des simples pages actives avec JHTML / JSP

1. Introduction à JHTML (Page Compilation)

Principe de base:

- Permet d'écrire des pages hybrides HTML / Java (comme PHP)
- Compile et exécute le source comme "servlet"
- La traduction se fait automatiquement pour tous les fichiers *.jhtml
  - le résultat (servlet source et classe) est placée dans le répertoire jws / pagecompile
  - Après chaque update d'un fichier *.jhtml la classe servlet est recompilée

Utilité:

- Création de pages "design"
- Analyse de formulaires
- possibilité d'interfaçage avec des "vrais" servlets

URL et documentation du serveur JAVA

- URL Serveur JAVA / TECFA: http:/ / tecfa2.unige.ch:8080 /
- Doc générale: http:/ / tecfa.unige.ch / guides / java / pointers.html
- Accès aux étudiants, voir le module act-java-jhtml
- **Doc JHTML: http://tecfa2.unige.ch/guides/java/java-web-server/page_comp/intro.html**
- Voir aussi 3.1 "Principe de base du traitement de requêtes" [7]

1.1 Ecrire des pages JHTML simples

- pas besoin d'importer les classes (c'est fait par le serveur!)
- Le code Java est délimité dans une page HTML par:
  - <java> ..... </java>
- Le code Java est délimité dans un TAG HTML par:
  - `‘...‘` (backquotes)
  - voir exemple 1-2 “JHTML simple (2)” [5]

Exemple 1-1: JHTML simple (1)

url: **Programme:** http://tecfa2.unige.ch:8080/develop/jhtml-ex/demo1.jhtml
url: **Source:** http://tecfa2.unige.ch:8080/develop/jhtml-ex/demo1.text

```
<BODY>
  <H1>JHTML Test</H1>
   <ul>
    <java>
    for (int i = 0; i < 5; i++) out.println ("<li>" + i);
    </java>
   </ul>
</BODY>
```

- `out` est un objet "gratuitement" mis à votre disposition pour les "sorties"
- Curieux: voir -6. “JHTML inside” [15] pour le Java que cette page génère

1.2 Faire une soupe HTML/Java plus mélangée

Exemple 1-2: JHTML simple (2)

url: **voir** http://tecfa2.unige.ch:8080/develop/jhtml-ex/demo2.jhtml
url: **(source)** http://tecfa2.unige.ch:8080/develop/jhtml-ex/demo2.text

Dans les sections Java, il faut quoter les " qui doivent appararaître dans HTML (\")

```
for (int i = 0; i < size; i++) {
  out.println ("<li>");
  if (i == emphasize) out.println ("<b>");
  out.println(" \"item" + i + ".jhtml\" - item number " + i);
  if (i == emphasize) out.println ("</b>");
}
```

On peut mélanger variables Java et strings HTML

- utiliser des back-quotes (‘)

```
<java>
int font_size = 20; String color = "red";
</java>

<font color="‘color‘" size="‘font_size‘"> Youppie ! </font>
```

Mais faites pas trop compliqué ....

- Faire des simples out.println et utiliser des + fait aussi l’affaire

2. Utilisation d'autres classes JAVA

- Vous pouvez importer toutes les classes Java, mais attention à la syntaxe simple: chaque import est mis sur une ligne séparée (sans mot clef "import" !!!)

```
<java type=import>
  import
  import
</java>
```

Exemple 2-1: Dates avec JHTML

url: http://tecfa2.unige.ch:8080/develop/jhtml-ex/weekday.jhtml
url: http://tecfa2.unige.ch:8080/develop/jhtml-ex/weekday.text

```
<java type=import>
java.util.Date
</java>
<html> <head><title>Today</title></head> <body> <h1>Today</h1>

<java>
// Maps day number to a name
String days[] = {"Dimanche","Lundi","Mardi","Mercredi","Jeudi","Vendredi",
"Samedi"};
// Get today's date
Date today = new Date ();
int weekday = today.getDay ();
out.println ("<p>On est " + days [weekday] + " aujourd'hui!");
</java> </body> </html>
```

3. Traitement de formulaires

3.1 Principe de base du traitement de requêtes

- Les pages JHTML peuvent utiliser les fonctionalités de la classe javax.servlet (puisqu'elles sont traduites en un servlet)
- Cela signifie notamment que le serveur crée automatiquement 2 objets:

1. `request` de la classe javax.servlet.http.HttpServletRequest
2. `out` de la classe javax.servlet.ServletOutputStream
3. `response` de la classe javax.servlet.http.HttpServletResponse

- Pour récupérer une variable "formulaire" utiliser:

```
request.getParameter("paramètre")
```

  - paramètre est le nom du paramètre qui vient des GET, POST, COOKIE, etc.
  - request possède d'autres méthodes utiles (!)

Il y a plus de documentation:

url: **http://tecfa2.unige.ch/guides/java/java-web-server/page_comp/intro.html** (intro)

url: **http://tecfa2.unige.ch/guides/java/jsdk2/doc/apidoc/packages.html** (servlet API)

- Quick reference:

url: **http://tecfa2.unige.ch/guides/java/java-web-server/page_comp/appB.html**

Exemple 3-1: Simple GET Demo.

url: **simple: http://tecfa2.unige.ch:8080/develop/jhtml-ex/request1.jhtml**
url: **argument: /develop/jhtml-ex/request1.jhtml?MESSAGE=Hello**
url: **source: http://tecfa2.unige.ch:8080/develop/jhtml-ex/request1.text**

```
<html>
<body>
<java>
String message;
    if ((message = request.getParameter("MESSAGE")) == null) {
      out.print("No message query argument supplied.");
      out.print("Please use an URL like request1.jhtml?MESSAGE=Hello");
    }
else
    out.print("Message is " + message);
</java>
</body> </html>
```

3.2 Traitement simple de formulaires

Exemple 3-2: Simple calcul (Formulaire + JHTML)

url: **http://tecfa2.unige.ch:8080/develop/jhtml-ex/simple-calcul/formulaire.html**

url: **source JHtml: /develop/jhtml-ex/simple-calcul/calcul.text**

Formulaire HTML:

```
<form action="calcul.jhtml" method=post>
Quelles sont vos connaissances de HTML ?
<input type="radio" name="choice" value="1" checked>faibles
<input type="radio" name="choice" value="2">moyennes
<input type="radio" name="choice" value="3">bonnes
<br>
Indiquez votre expertise en programmation:
<input type="radio" name="choice2" value="1" checked>absente
<input type="radio" name="choice2" value="2">moyenne
<input type="radio" name="choice2" value="3">bonne
<P>
<input type="submit" value="Voir le résultat!">
</form>
```

3.3 Vérification de l'input

Voir si un paramètre est vide:

1. Champs texte
   - on regarde s'ils sont vides (string de taille zéro)
   - un input type "text" (HTML) est donc toujours passé au serveur !!

```
// un string vide pour comparer
String emptyString = new String ("");
// input
String AdresseString = req.getParameter("adresse");
.....
if  ( AdresseString.equals(emptyString) ) { ... geuler ...}
```

2. Widgets de type "radio" etc.
   - on regarde si l'objet est présent ou absent (le formulaire ne transmet rien)
   - si "choix" n'a pas été transmis, getParameter retourne `null` !

```
String Choix = req.getParameter("choix");
....
if (Choix == null) { .... geuler .... }
```

JAVA:

```
// Parameters come as strings
String choice = request.getParameter("choice");
String choice2 = request.getParameter("choice2");

// Integer.parseInt() translates a string to an Integer
int score = Integer.parseInt(choice) + Integer.parseInt(choice2);

out.print("<h3>Votre score est de " + score + "</h3>");

if (score < 3) {
  out.print ("<p>Vous êtes un débutant</p>");
} else if (score < 5) {
  out.print ("<p>Vous avez un niveau moyen</p>");
} else {
  out.print ("<p>Vous êtes un expert !</p>");
}
```

- la méthode request.getParameter retourne la valeur d'une variable
  - sous forme de string, il faut donc déclarer choice comme String
- la méthode Integer.parseInt("string") traduit un string en entier
  - nécessaire pour faire une addition dans l'exemple ci-dessus.

Pour les tricheurs, un emploi "GET" de la page:

url: **/develop/jhtml-ex/simple-calcul/calcul.jhtml?choice=10&choice2=15**

4. HTML conditionnel

- On peut afficher conditionellement du HTML
- ATTENTION: pas aussi simple que PHP
  - parfois ca peut générer des lignes qui font sauter un if ... then ...else
  - si besoin aller voir le code java généré dans /pagecompile/....
    (voir section 6. “JHTML inside” [15])
  - en cas de doute: utiliser "out.print ..... " à la place.

Exemple 4-1: Simple HTML conditionnel

- voir: **http://tecfa2.unige.ch:8080/develop/jhtml-ex/html-mix.jhtml**

```
<java>
double balance = 140.25;
</java>
<p>Your current balance is: <java> out.print (balance);
if (balance >= 100.00) { </java>
<p>You use the phone far too much!
<java> } </java>
</body> </html>
```

- Le tag </java> est à l’INTERIEUR de la clause if:

```
<java> if (balance >= 100.00) { </java>
```

- Même chose pour fermer cette clause:

```
<java> } </java>
```

Exemple 4-2: Simple calcul en une seule page JHTML

url: **http://tecfa2.unige.ch:8080/develop/jhtml-ex/simple-calcul/formulaire2.jhtml**
url: **http://tecfa2.unige.ch:8080/develop/jhtml-ex/simple-calcul/formulaire2.text**

- L' exemple 3-2 "Simple calcul (Formulaire + JHTML)" [9] en une page...

- En règle générale il semble plus simple d'utiliser 2 pages
  - une pour le formulaire et une autre pour le traitement.
- Voir les "names" / variables process et process2
  - d'abord jouer avec l'exemple avant de comprendre la logique ci-dessous

```
<java>
if ((process != null) || (process2 !=null)) {
   ... montrer les résultats
   if (process2 != null) {
   ... afficher un merci
  }
    else {
    ... dire que l'on peut essayer de nouveau
    }
}
if (process2 == null)  {
</java>
   ... afficher le formulaire
<input type="submit" name="process" value="Voir le résultat!">
<input type="submit" name="process2" value="Voir le résultat et finir!">
<java> } </java>
```

5. Session tracking

url: **http://tecfa2.unige.ch/guides/java/java-web-server/session_track/SessionTr.html**

Exemple 5-1: Session tracking et dates

url: http://tecfa2.unige.ch:8080/develop/jhtml-ex/session-track.jhtml
url: http://tecfa2.unige.ch:8080/develop/jhtml-ex/session-track.text

```
<java type=import>
javax.servlet.http.*
java.text.*
</java>
<java>
HttpSession session = request.getSession (true);

// Get the session data value
Integer ival = (Integer) session.getValue ("sessiontest.counter");
if (ival == null) ival = new Integer (1);
else ival = new Integer (ival.intValue () + 1);
session.putValue ("sessiontest.counter", ival);
</java>

You have hit this page <java type=print>ival</java> times.<br>
<java>
out.println("Your Session ID is " + session.getId() +  "<br>");
</java>
```

6. JHTML inside

- Section à option (à ce stade)

  Il n'est pas nécessaire de comprende le code java produit pour utiliser les pages jhtml. Par contre il est utile de savoir lire une telle page approximativement pour comprendre des messages d'erreur

Mécanisme

- Le server Java met les fichiers *.java et *.class générés dans le répertoire /local/jws/jws/pagecompile (à Tecfa)
- Pour chaque page JHTML, un "servlet" est créé:
  - dans l'exemple: _demo1
  - il s'agit d'une sous-classe de HttpServlet
- Une seule méthode "service" est crée
  - elle incorpore les instructions java dans la page jhtml
  - elle "copie" plus ou moins "tel quel" les commandes html (voir méthode writeBytes)
- Les classes *.webserver.pagecompile assurent la mise à jour de la classe dès que le code source *.jhtml change.

Sécurité

- Les servlets générés par JHTML sont "sandboxed":
  - pas d'accès aux fichiers et autres ressources locales
  - (Sauf si je me trompe) il faut soit écrire des vrais servlets pour cela.

Exemple 6-1: Java généré pour l'exemple 1-1 "JHTML simple (1)" [4]

```
package pagecompile._develop._jhtml_sex;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.sun.server.webserver.pagecompile.filecache.*;
import com.sun.server.webserver.pagecompile.ParamsHttpServletRequest;
import com.sun.server.webserver.pagecompile.*;
public class _demo1 extends HttpServlet {

  static {  }

    //-------------- The service method
  public void service (HttpServletRequest request,
                       HttpServletResponse response)
      throws ServletException, IOException
  {
    ServletOutputStream out = response.getOutputStream ();
    ByteFileData __fileData = null;
    try {
      __fileData = (ByteFileData) ServletUtil.getJHtmlSource(this, "/local/servers/jws/
JavaWebServer1.1.1/public_html/develop/jhtml-ex/demo1.jhtml", null, 912504010000L);
      if (__fileData == null) throw new ServletException("FileChanged");
```

```
      /*** lines: 1-15 */
      __fileData.writeBytes (0, 264, out);
      for (int i = 0; i < 5; i++) out.println ("<li>" + i);
            /*** lines: 17-17 */
      __fileData.writeBytes (346, 57, out);
      out.print(ServletUtil.encodeURL (request, response, "http://tecfa.unige.ch/tecfa-people/
schneider.html"));
      /*** lines: 17-26 */
      __fileData.writeBytes (452, 46, out);
    }
    finally {
      if (__fileData != null) __fileData.close();
    }
  }
}
```