Java CGI HOWTO

David H. Silber *dhs@orbits.com* . v0.4, 18 Novembre 1996.

Ce document vous explique comment convaincre votre serveur WWW d'utiliser des programmes de CGI écrits en Java, et montre l'emploi de Java dans l'écriture de programmes de CGI. Bien que les HOWTO se restreignent en principe au système Linux, celui-ci est indépendant de la version d'Unix utilisée. *Traduction :* Xavier Cazin , *le 27 novembre 1997*.

Contents

1 Introduction

Par construction du language, les variables d'environnement du système ne sont pas facilement accessibles au programmeur Java. Par ailleurs, le JDK (Java Development Kit) rend impossible l'invocation directe d'un programme, ce qui ne facilite pas le traitement standard par CGI des formulaires HTML. On peut contourner ces limitations de plusieurs façons. Vous saurez comment j'ai implémenté l'une d'elles en poursuivant votre lecture.

1.1 Pré-requis

Je considère que vous êtes familiarisé avec les principes qui sous-tendent HTML et CGI, et que vous possédez un minimum de connaissances de votre serveur HTTP. La programmation Java ne devra pas non plus vous être étrangère, à défaut de quoi ce qui va suivre ne vous sera pas très parlant.

1.2 Ce document

http://www.orbits.com/software/Java_ CGI.html est l'adresse où vous êtes sûr de trouver la dernière version de ce document.

1.3 Le package lui-même

L'archive *ftp://ftp.orbits.com/pub/software/java_ cgi-0.4.tgz*

contient la dernière version du package décrit ici. Vous y trouverez également le source SGML de ce document (en anglais bien sûr).

La distribution du package suit les recommandations de la LGPL (GNU Library General Public License). Ce document peut être distribué selon les termes gouvernant le copyright des Linux HOWTO.

Merci de bien vouloir mentionner le document *http://www.orbits.com/software/Java_CGI.html* si vous utilisez ce logiciel. Vous permettrez ainsi à d'autres d'accéder aux classes Java CGI.

1.4 Publicité gratuite

Ce document a été mis au point avec la bienveillance de **Stellar Orbits Technology Services**. Si vous voulez savoir ce que nous faisons, allez voir à *http://www.orbits.com/* .

2 Configuration de votre serveur (avec explications)

Cette section vous conduira à travers l'installation de mon package *Java CGI*, et sera agrémentée d'explications généreuses qui vous permettront de mesurer les conséquences de vos actes. Si vous souhaitez simplement installer les programmes, sans vous soucier du pourquoi et du comment, sautez directement à la section 3 (Configuration du serveur (version courte)).

2.1 Contraintes logicielles et matérielles

Ce logiciel devrait fonctionner sur n'importe quel système à la Unix sur lequel se trouvent au moins installés le JDK et un serveur Web. J'utilise pour ma part un *Linux Debian* sur lequel tourne le démon HTTP *apache*. Si cela ne fonctionne pas sur votre installation, n'hésitez pas à me contacter à *dhs@orbits.com* .

Malheureusement, l'interpréteur Java n'est pas particulièrement économe en mémoire ; si vous devez utiliser souvent des programmes de CGI en Java, quelques mégaoctets de RAM supplémentaires ne seront pas de trop.

2.2 Java CGI

Le logiciel que j'ai écrit s'appelle *Java CGI* (Note: au cas où vous ne l'auriez pas encore remarqué (NdT)). Vous pouvez le récupérer par ftp anonyme à l'adresse *ftp://www.orbits.com/pub/software/java_cgi-0.4.tgz* . (Le numéro de version peut avoir changé.)

2.3 Déploiement des sources

Choisissez un répertoire où vous pourrez tranquillement déployer l'archive du package. Je suggère généralement `/usr/local/src`. Désarchivez ensuite à l'aide de la commande (Note : les “lignuxeurs” préfèreront sans doute le plus élégant `tar xzvf java_cgi-0.4.tgz` (NdT).) :

```
gzip -dc java_cgi-0.4.tgz | tar -xvf -
```

Cela aura pour effet de créer un répertoire de nom `java_cgi-0.4`. Vous y trouverez les fichiers auxquels nous feront référence dans la suite. (Si le numéro de version a changé, suivez les instructions qui s'y trouvent à partir de maintenant).

2.4 Chemins locaux

Vous allez devoir décider de l'endroit où vous souhaitez que les programmes Java CGI résident. La plupart du temps, vous aurez intérêt à les placer dans un répertoire parallèle au répertoire `cgi-bin`. La configuration de mon serveur *apache* indiquait `/var/web/cgi-bin` comme répertoire `cgi-bin` par défaut. J'ai donc placé mes programmes Java CGI dans le répertoire `/var/web/javacgi`. Il n'est pas conseillé de placer ces programmes dans l'un des répertoires référencés par `CLASSPATH`. Éditez le Makefile pour refléter la configuration de votre système. En tant qu'utilisateur root, lancez `make install`. Cela aura pour effet de compiler vos programmes Java, modifier le script `java.cgi` pour qu'il s'adapte à votre système, et installer les programmes au bon endroit. Si vous souhaitez également disposer d'une version HTML de ce document, et d'un document test en HTML, lancez plutôt `make all`.

2.5 Test de votre installation

Les documents `javacgitest.html`, `javaemailtest.html` et `javahtmltest.html` devraient maintenant être installés. Si vous avez choisi `make all`, ils se trouveront dans le répertoire spécifié par la variable WEBDIR du `Makefile`. Dans le cas contraire, vous pouvez lancer `make test` pour les créer à partir de `javacgitest.html-dist`, `javaemailtest.html-dist` et `javahtmltest.html-dist`.

Après vous être assuré que votre installation s'était déroulée correctement, vous pouvez supprimer les fichiers `CGI_Test.class`, `Email_Test.class` et `HTML_Test.class` de votre répertoire JAVACGI, ainsi que `javacgitest.html`, `javaemailtest.html` et `javahtmltest.html` de votre répertoire WEBDIR. Ils montrent les informations utilisateurs auxquelles le serveur est normalement seul à avoir accès.

3 Configuration du serveur (version courte)

- Récupérez le package *Java CGI* à partir de *ftp://www.orbits.com/pub/software/java_ cgi-0.4.tgz* . (Le numéro de version peut avoir changé.)
- Déployez la distribution à l'aide de la commande

  ```
  gzip -dc java_cgi-0.4.tgz | tar -xvf -
  ```

  (Si le numéro de version de la distribution a changé, utilisez les instructions qui s'y trouvent à partir de maintenant.)
- Éditez le `Makefile` que vous trouverez dans le nouveau répertoire `java_cgi-0.4` pour qu'il reflète la configuration de votre système.
- En tant que root, lancez `make install`. Cela aura pour effet de compiler les programmes Java, prendre en compte les informations propres à votre système, et installer les divers fichiers.

  Si vous souhaitez disposer d'une version HTML de ce document, ainsi que d'un document test en HTML, lancez plutôt `make all`.
- Vous devriez maintenant être paré.

4 Exécution d'un programme Java CGI

4.1 Difficultés d'exécution de programmes Java avec le modèle CGI

L'exécution d'un programme Java depuis un serveur Web pose deux types de problèmes majeurs :

4.1.1 Les programmes Java ne s'exécutent pas comme des binaires ordinaires

Il faut lancer l'interpréteur Java et fournir la classe principale (le programme à exécuter) sur la ligne de commande. Les formulaires HTML ne permettent pas d'envoyer directement une ligne de commande au serveur Web.

4.1.2 Java n'accède pas *a priori* aux variables d'environnement

Toutes les variables d'environnement requises par le programme Java doivent lui être passées explicitement. Il n'existe pas de méthode similaire à la fonction `getenv()` de **C** .

4.2 Solutions proposées

Pour contourner ces obstacles, j'ai écrit une script shell de CGI, qui fournit les informations nécessaires à l'interpréteur Java.

4.2.1 Le script java.cgi

Ce script de shell se charge de l'interaction entre le démon HTTP et le programme Java CGI que vous souhaitez utiliser. Il extrait le nom du programme que vous souhaitez lancer à partir des données fournies par le serveur. Il récupère ensuite toutes les valeurs d'environnement dans un fichier temporaire. Enfin, il lance l'interpréteur Java en lui passant le nom du fichier contenant les informations d'environnement, ainsi que le nom du programme à exécuter.

Le script `java.cgi` a été configuré et installé selon les procédure décrites à la section 2.4 (Decide On Your Local Path Policies).

4.2.2 Invocation de java.cgi depuis un formulaire HTML

Mes formulaires qui utilisent les programmes Java CGI spécifient l'action à effectuer de la façon suivante :

```
<form action="/cgi-bin/java.cgi/CGI_Test" method="POST">
```

où `/cgi-bin/` est votre répertoire local d'exécutables CGI, `java.cgi` est l'interface permettant de lancer les programmes Java, et `CGI_Test` est un exemple de programme Java à exécuter.

5 Utilisation des classes Java CGI

Trois classes principales sont pour l'instant supportées : 5.1 (CGI), 5.3 (Email) et 5.5 (HTML). Je pense y ajouter des classes capables de gérer des entrées et des sorties formatées en MIME (respectivement MIMEin & MIMEout).

Quelques classes de test et de support sont également disponibles : 5.2 (CGI_Test), 5.4 (Email_Test) et 5.4 (HTML_Test) doivent permettre de tester votre installation. Elles peuvent aussi servir de point de départ à vos propres programmes Java basés sur cette bibliothèque de classes. La classe 5.7 (Text) est une superclasse des classes `Email` et `HTML`.

5.1 CGI

5.1.1 Syntaxe

```
public class CGI
```

5.1.2 Description

La classe CGI détient les "informations SGI" : les valeurs d'environnement initialisées par le serveur Web ainsi que le nom et la valeur issus du formulaire quand l'action **submit** est sélectionnée. Toutes les informations sont stockées dans un objet de classe `Properties`.

Cette classe se trouve dans le package "Orbits.net".

5.1.3 Liste des membres

```
CGI()         //  Constructeur.
getNames()    //  Recupere la liste de noms.
getValue()    //  Recupere la valeur a partir du nom.
```

5.1.4 Voir aussi

`CGI_Test`.

5.1.5 CGI()

Finalité

Construit un objet contenant les données CGI disponibles.

Syntaxe

`public CGI()`

Description

Lorsqu'un objet CGI est construit, toutes les informations CGI disponibles sont récupérées et stockées dans le nouvel objet.

5.1.6 getNames()

Finalité

Dresse la liste des noms définis par le formulaire.

Syntaxe

`public Enumeration getNames ()`

Description

Fournit la liste complète des noms pour lesquels des valeurs correspondantes ont été définies.

Retourne

Une `Enumeration` de tous les noms définis.

5.1.7 getValue()

Finalité

Récupère la **valeur** associée au **nom** spécifié.

Syntaxe

```
public String getValue ( String nom )
```

Description

Cette méthode fournit la correspondance entre les `noms` et les `valeurs` envoyées depuis un formulaire HTML.

Paramètre

nom

La clé par laquelle les valeurs sont choisies.

Retourne

Une `String` contenant la valeur.

5.2 CGI_Test

Cette classe fournit à la fois un exemple d'utilisation de la classe CGI, et un programme de test, qu'on pourra utiliser pour confirmer que le package *Java CGI* fonctionne correctement.

5.2.1 Liste des membres

```
main()      //   main() du programme
```

5.2.2 Voir aussi

`CGI`.

5.2.3 main()

Finalité

Fournit une méthode `main()`.

Syntaxe

```
public static void main( String argv[] )
```

Description

Il s'agit du point d'entrée d'un programme CGI qui ne fait rien à part retourner la liste des couples nom/valeur.

Paramètre

argv[]

Arguments passés au programme par le script `java.cgi`. Non utilisé pour l'instant.

5.3 Email

5.3.1 Syntaxe

```
public class Email extends Text
```

5.3.2 Description

Les messages sont construits au moyen des méthodes `add*()` de la classe `Text` et les méthodes spécifiques au courrier électronique fournies par cette classe. Une fois composé, le message est envoyé vers sa destination finale.

Cette classe se trouve dans le package "Orbits.net".

5.3.3 Liste des membres

```
Email()      //  Constructeur
send()       //  Envoie le message e-mail
sendTo()     //  Ajoute une destination au message
subject()    //  Initialise le champ Subject: du message
```

5.3.4 Voir aussi

`Email_Test`, `Text`.

5.3.5 Email()

Finalité

Construit un objet qui contiendra un message électronique.

Syntaxe

```
public Email()
```

Description

Crée un message vide qui sera rempli par les méthodes Email.

Voir aussi

`Text`.

5.3.6 send()

Finalité

Envoie le message e-mail.

Syntaxe

```
public void send ()
```

Description

Formatage et envoi du message. Si aucune adresse de destination n'a été précisée, ne fait rien.

5.3.7 sendTo()

Finalité

Ajoute une destination pour ce message.

Syntaxe

```
public String sendTo ( String adresse )
```

Description

Ajoute `adresse` à la liste des destinations pour cette méthode. Il n'existe pas de limite *a priori* pour le nombre de destinations d'un message e-mail. Je suis sûr qu'avec une liste assez grande, on peut dépasser la taille acceptable pour le *Mail Transport Agent*, voire la mémoire disponible sur votre système.

Paramètre

adresse
Une destination à laquelle envoyer ce message.

5.3.8 subject()

Finalité

Initialise le sujet du message.

Syntaxe

```
public void subject ( String sujet )
```

Description

Cette méthode remplit le champ `Subject:` du message. Si elle est appelée plusieurs fois, le sujet utilisé sera le dernier demandé.

Paramètre

sujet
Le texte du champ `Subject:` du message.

5.4 Email_Test

Cette classe fournit à la fois un exemple d'utilisation de la classe `Email` et un programme de test qu'on pourra utiliser pour s'assurer que le package *Java CGI* fonctionne correctement.

5.4.1 Liste des membres

```
main()      //  main() du programme
```

5.4.2 Voir aussi

`Email`.

5.4.3 main()

Finalité

Fournit une méthode `main()`.

Syntaxe

```
public static void main( String argv[] )
```

Description

Il s'agit du point d'entrée d'un programme CGI qui retourne une liste des couples nom/valeur disponibles. Cette liste sera également envoyée à l'adresse spécifiée dans la variable `Email`.

Paramètre

argv[]

Arguments passés au programme par le script `java.cgi`. Non utilisé pour l'instant.

5.5 HTML

5.5.1 Syntaxe

```
public class HTML extends Text
```

5.5.2 Description

Les messages sont créés à l'aide des méthodes `add*()` de la classe `Text` et des méthodes spécifique au HTML ajoutées par cette classe. Une fois terminé, le message est envoyé.

Aucun test n'est effectué pour l'instant pour s'asurer que les méthodes de construction de liste sont utilisées dans le bon ordre. C'est donc au programmeur de faire attention à ne pas violer la syntaxe HTML.

Cette classe se trouve dans le package “Orbits.net”.

5.5.3 Liste des membres

```
HTML()                  //  Constructeur.
author()                //  Initialise le nom de l'auteur du document.
definitionList()        //  Cree une liste de definitions.
definitionListTerm()    //  Ajoute un terme a la liste de definitions.
endList()               //  Termine une liste.
listItem()              //  Ajoute une entree a une liste.
send()                  //  Envoie le message HTML.
title()                 //  Initialise le titre du document.
```

5.5.4 Voir aussi

`HTML_Test`, `Text`.

5.5.5 HTML()

Finalité

Construit un objet qui contiendra un message HTML.

Syntaxe

```
public HTML()
```

Description

Crée un message vide qui sera rempli par les méthodes HTML.

Voir aussi

`Text`.

5.5.6 author()

Finalité

Initialise le nom de l'auteur du document.

Syntaxe

```
public void author ( String auteur )
```

Description

Donne au document un nom d'auteur ayant pour valeur `author`.

Paramètre

auteur
Texte à utiliser en tant que nom d'auteur du message.

Voir aussi

`title()`.

5.5.7 definitionList()

Finalité

Crée une liste de définitions.

Syntaxe

```
public void definitionList ()
```

Description

Initialise une liste de définition. Une *liste de définitions* est une liste spécialisée telle que chaque entrée de la liste soit un *terme* suivi du *texte* correspondant à la définition de ce terme. La création d'une liste de définitions doit être suivie par celle d'au moins un couple terme/texte, et d'un appel à la méthode `endList()`. *Notons que, pour le moment, les listes ne peuvent pas être imbriquées.*

Voir aussi

`definitionListTerm()`, `endList()`, `listItem()`.

5.5.8 definitionListTerm()

Finalité

Ajoute un terme à la liste de définitions.

Syntaxe

```
public void definitionListTerm ()
```

Description

Ajoute un terme à la liste de définitions. Le texte définissant le partie terme de l'entrée courante de la liste devra être inséré dans le message après l'appel de cette méthode, et avant qu'une méthode `listItem` correspondante soit appelée.

Voir aussi

`definitionList()`, `listItem()`.

5.5.9 endList()

Finalité

Termine une liste.

Syntaxe

```
public void endList ()
```

Description

Cette méthode permet de clore une liste. *Notons que, pour le moment, les listes ne peuvent pas être imbriquées.*

Voir aussi

`definitionList()`.

5.5.10 listItem()

Finalité

Ajoute une entrée à une liste.

Syntaxe

```
public void listItem ()
```

```
public void listItem ( String article )
```

```
public boolean listItem ( String terme, String article )
```

Description

Ajoute une entrée à une liste. Si la première forme est utilisée, le texte de l'article de liste courant devra être ajouté au message après l'appel de cette méthode, et avant tout autre appel à des méthodes de liste. Dans la deuxième et troisième forme, le texte `article` est passé comme paramètre à la méthode, au lieu (ou en plus) d'être ajouté au message. La troisième forme est spécifique aux listes de définitions et fournit à la fois le terme et la définition de l'entrée de liste.

Paramètres

article
Le texte de cette entrée de liste de définitions.

terme
Le texte de la partie terme de cette entrée de liste de définitions.

Voir aussi

`definitionList()`, `definitionListTerm()`, `endList()`.

5.5.11 send()

Finalité

Envoie le message HTML.

Syntaxe

```
public void send ()
```

Description

Envoie le message HTML.

5.5.12 title()

Finalité

Donne une valeur au titre du document.

Syntaxe

```
public void title ( String titre )
```

Description

Initialise le texte du titre du document.

Paramètre

titre
Le texte du titre de ce message.

Voir aussi

`author()`.

5.6 HTML_Test

Cette classe offre à la fois un exemple d'utilisation de la classe `HTML` et un programme de test qui peu servir à s'assurer que le package *Java CGI* fonctionne correctement.

5.6.1 Liste des membres

```
main()      //  main() du programme.
```

5.6.2 Voir aussi

`HTML`.

5.6.3 main()

Finalité

Fournit une méthode `main()`.

Syntaxe

```
public static void main( String argv[] )
```

Description

Il s'agit du point d'entrée pour un programme CGI qui retourne une liste des couples nom/valeur d'un document HTML, chaque couple étant un élément d'une liste de définitions.

Paramètre

argv[]

Arguments passés au programme par le script `java.cgi` Non utilisé pour l'instant.

5.7 Text

5.7.1 Syntaxe

```
public abstract class Text
```

5.7.2 Description

Cette classe est la superclasse des classes `Email` et `HTML`. Les messages sont construits à l'aide des méthodes de cette classe, puis complétés et formatés grâce aux méthodes des sous-classes.

Cette classe se trouve dans le package "Orbits.text".

5.7.3 Liste des membres

```
Text()             //  Constructeur.
add()              //  Ajoute du texte a cet objet.
addLineBreak()     //  Ajoute une rupture de ligne.
addParagraph()     //  Ajoute une rupture de paragraphe.
```

5.7.4 Voir aussi

`Email`, `HTML`.

5.7.5 add()

Finalité

Ajoute du texte à cet article

Syntaxe

```
public void add ( char ajout )
```

```
public void add ( String ajout )
```

```
public void add ( StringBuffer ajout )
```

Description

Ajoute le texte `ajout` à la suite du contenu de cet article.

Paramètre

ajout
Texte à ajouter.

Voir aussi

`addLineBreak()`, `addParagraph()`.

5.7.6 addLineBreak()

Finalité

Force une rupture de ligne à cet endroit dans le texte.

Syntaxe

```
public void addLineBreak ()
```

Description

Insère une rupture de ligne dans le texte, à l'endroit du point courant.

Voir aussi

`add()`, `addParagraph()`.

5.7.7 addParagraph()

Finalité

Débute un nouveau paragraphe.

Syntaxe

```
public void add ()
```

Description

Débute un nouveau paragraphe à ce point du flot textuel.

Voir aussi

`add()`, `addLineBreak()`.

6 Améliorations prévues

- Ajouter à la classe Email :

  Email(int capacité)
  : Quand on connaît à l'avance l'espace à alloué au message.

  sendTo(String [] adresse)
  : Ajoute une liste de destinations principales (champ `To:`) au message e-mail.

  sendCc(String adresse)
  : Ajoute une destination au champ `Cc:` (Carbon-Copy) du message e-mail.

  sendCc(String [] address)
  : Ajoute une liste de destinations au champ `Cc:` (Carbon-Copy) du message e-mail.

  sendBcc(String adresse)
  : Ajoute une destination au champ `Bcc:` (Blind Carbon-Copy) du message e-mail.

  sendBcc(String [] adresse)
  : Ajoute une liste de destinations au champ `Bcc:` (Blind Carbon-Copy) du message e-mail.

- Ajouter à la classe HTML :

  HTML(int capacité)
  : Quand on connaît à l'avance l'espace à alloué au message.

  public void unorderedList()
  : Démarre une liste non ordonnée.

  public void orderedList()
  : Démarre une liste ordonnée.

  public void directoryList()
  : Démarre une liste de répertoires.

  public void menuList()
  : Démarre une liste de menu.

  void anchor(String anchorName)
  : Spécifie une ancre.

  void link(String url, String text)
  : Spécifie un lien.

  void applet(String url, String altText)
  : Spécifie un lien applet.

- Autoriser l'imbrication des listes HTML.
- Ajouter du code de gestion d'erreur pour s'assurer que le formatage des listes HTML s'effectue dans le bon ordre.
- L'emplacement du fichier des informations d'environnement devra être configurable à partir du `Makefile`.
- Éliminer les couples nom/valeur vides qui apparaissent dans la liste quand la méthode GET est utilisée pour le transfert des données.
- Réfléchir à un moyen d'implémenter l'interface java.util.Enumeration avec CGI pour fournir des noms de variables.

- Ajouter une classe `Test`, qui permettrait d'utiliser toutes les méthodes de ce package.
- Documenter la manière dont `CGI_Test`, `Email_Test` et `HTML_Test` se réfèrent mutuellement pour fournir des tests incrémentaux facilitant le débogage.
- Documenter la manière dont Test se sert de toutes les fonctionnalités présentes dans le package.

7 Modifications

7.1 Modifications entre les versions 0.3 et 0.4

- Développement de la classe HTML pour qu'elle fournisse des fonctionnalités minimales.
- Écriture de la classe HTML_Test et de javahtmltest.html-dist.
- Ajout des méthodes HTML gérant les listes de définitions.

7.2 Modifications entre les versions 0.2 et 0.3

- Ajout des classes Text et Email. HTML a été également ajouté, sans qu'il soit vraiment utilisable.
- Placement des diverses classes dans des packages. Les classes principales se trouvent dans `Orbits.net.*`, la classe de support `Text` se trouve dans `Orbits.text.Text`.
- `CGItest` devient `CGI_Test`.
- Ajout de la classe `Email_Test`.

7.3 Modifications entre les versions 0.1 et 0.2

- Les variables d'environnement sont placées dans un fichier temporaire, au lieu d'être passées à l'interpréteur Java sur la ligne de commande. La classe `CGI` et le script `java.cgi` durent être modifiés.
- Le document `javacgitest.html` devient partie intégrante de la distribution.
- Les fichiers texte qui sont modifiés par `make` lors de l'installation sont fournis avec des noms terminés par *-dist*.