BeanShellEditor

Version 0.1.1

Michaël MICHAUD

BeanShellEditor: Version 0.1.1

Michaël MICHAUD

BeanShellEditor est un éditeur de script pour BeanShell [http://www.beanshell.org/home.html]. Il peut être utilisé comme application autonome ou comme PlugIn d'une autre application. Dans les deux cas, il permet de réaliser de petits programmes très facilement, d'accéder à un script existant en deux clicks (script manager), d'import facilement des nouveaux packages... Grâce au célèbre langage de script BeanShell, BeanShellEditor met la puissance de java entre les mains des non programmeurs. L'interface utilise l'excellente bibliothèque Buoy, et l'éditeur utilise le package JEditSyntax, ancienne version du fameux éditeur JEdit.

Le fonctionnement du programme nécessite la présence des bibliothèques suivantes :

- BeanShell, de Pat Niemeyer (bibliothèque LGPL)
- Buoy, de Peter Eastman (domaine publique)
- JEditSyntax (licence MIT license)

Table des matières

Chapitre 1. Introduction

1. Qu'est-ce que BeanShellEditor

BeanShell Editor est un éditeur de scripts pour BeanShell. BeanShell est un petit interpréteur de code source java gratuit, lui-même écrit en java, et qui possède les caractéristiques des langages de script orientés objet. Its mascott is  .L'auteur de cette bibliothèque est Pat Niemeyer. Vous trouverez le fichier binaire (jar), les sources et la documentation sur le site beanshell [http://www.beanshell.org]. BeanShellEditor est juste un éditeur facilitant l'usage de ce langage (vous trouverez également JConsole dans la distribution BeanShell, une application qui sert à peu près le même but que BeanShellEditor, mais qui ne répondait pas exactement à mon attente).

2. Comment installer BeanShellEditor

Pour installer BeanShellEditor, vous avez besoin de :

- un ordinateur avec un machine virtuelle java (jvm 1.4+) installée (téléchargeable à java.sun.com [http://java.sun.com])
- le fichier bsheditor.jar, qui contient l'application (et inclut JEditSyntax)
- la bibliothèque beanshell (également téléchargeable à www.beanshell.org [http://www.beanshell.org])
- la bibliothèque buoy (également téléchargeable à buoy.sourceforge.net [http://buoy.sourceforge.net])

La distribution de BeanShellEditor n'inclut pas forcément toutes les bibliothèques que vous pouvez être amenés à télécharger depuis leur site respectif (il s'agit de petites bibliothèques, < 1Mo)

2.1. Installer BeanShellEditor comme application autonome

Si `BSHEDITOR` est votre répertoire d'installation, vous devrez avoir :

```
+BSHEDITOR
   BeanShellEditor.bat (launcher for windows)
   +LIB
      bsheditor.jar
      beanshell.jar
      buoy.jar
```

Si `BeanShellEditor.bat` n'est pas présent dans votre distribution, vous devrez simplement créer un fichier contenant la ligne suivante :

```
java -cp "LIB/bsheditor.jar;beanshell.jar;buoy.jar"
         fr.michaelm.bsheditor.BeanShellEditor
```

ou

```
start java -cp "LIB/bsheditor.jar;beanshell.jar;buoy.jar"
      fr.michaelm.bsheditor.BeanShellEditor
```

si vous voulez cacher la console DOS après le lancement de la JVM.

Pour lancer l'application, double-cliquez sur BeanShellEditor.bat.

Sur une configuration Windows récente, vous devriez également pouvoir lancer BeanShellEditor en double-cliquant sur le fichier .jar de ce nom, après avoir placé les 3 fichiers jar (BeanShellEditor, BeanShell et Buoy) dans un même répertoire.

Les expériences d'utilisation de BeanShellEditor sous Linux, Mac ou autres sont les bienvenues.

2.2. Installer BeanShellEditor en tant que PlugIn

Installer BeanShellEditor en tant que PlugIn d'une autre application est très simple :

Placez `bsheditor.jar`, `beanshell.jar` et `buoy.jar` dans votre classpath.

Créez un bouton ou un élément de menu dans votre application, qui crée une nouvelle instance de BeanShellEditor.

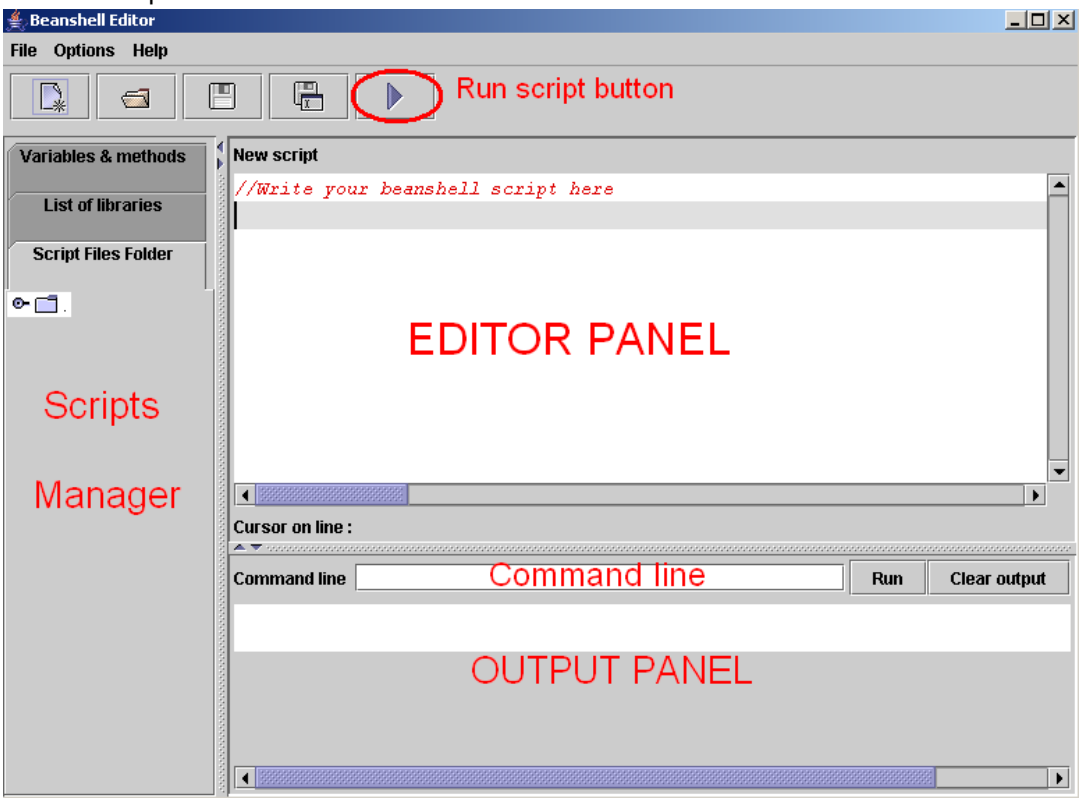
Vous devez utiliser le constructeur qui prend une table (Map) en paramètre. La table permettra de passer à l'éditeur un ou des noms de variables (les clés de la table) pointant sur des objets de votre application (les valeurs la table).

Par exemple, (méthode appelée par votre application pour créer une instance de BeanShellEditor.

```
public void createBeanShellEditor() {
   Map map = new HashMap();
   map.put("MyApplication", this); // MyApplication can be used
                                   // in your script editor to
                                   // access your application
   map.put("TA", textArea);        // TA can be used to access
                                   // the textArea attribute defined
                                   // in your MyApplication instance
   BeanShellEditor bsh = new BeanShellEditor(map);
}
```

Chapitre 2. Fonctionnalités de BeanShellEditor

Voici une copie d'écran de BeanShellEditor.



1. Le menu et la barre d'outils

Voici les commandes que vous trouverez dans le menu et/ou dans la barre d'outils :

1.1. Menu fichier

Le menu fichier inclut :

- New file : pour éditer un nouveau script. Vous pouvez aussi utiliser le bouton de la barre d'outils.
- Open file : pour ouvrir un fichier existant. Vous pouvez aussi utiliser le bouton de la barre d'outils ou encore le gestionnaire de scripts (voir section script manager)
- Save file : pour sauvegarder le fichier courant. Vous pouvez aussi utiliser le bouton de la barre d'outils.
- Save as : pour enregistrer le fichier courant sous un autre nom. Vous pouvez aussi utiliser le bouton de la barre d'outils.

1.2. Menu options

Les options de BeanShellEditor incluent :

- Choose scripts folder : pour choisir le répertoire où vous allez ranger vos scripts. BeanShellEditor représente ce répertoire sous forme d'une arborescence dans le gestionnaire de scripts. Ce dernier vous permet d'accéder à vos scripts d'un double-click.
- Choose jars folder : vous pouvez choisir un répertoire contenant des bibliothèques java que vous voulez utiliser dans vos scripts BeanShell. Dans le gestionnaire de script, vous pouvez choisir le panneau jars folder qui liste les fichiers jars contenus dans le répertoire. Un click droit sur le nom d'une des bibliothèques vous permet d'ajouter la bibliothèque au classpath de votre script (`addClassPath`) et d'importer les packages de votre choix (`import`).
- Choose start file : permet de choisir le script de démarrage (voir aussi ci-dessous). Le script de démarrage concerne uniquement l'éxécution des scripts à partir de l'éditeur. Il n'est pas relancé avant l'exécution d'une ligne de commande.
- Always execute start file : si cette option du menu est cochée, le script de démarrage sera toujours exécuté avant l'exécution d'un script lancé à partir de l'éditeur.
- Verbose outputs : si cette option est cochée, vous obtiendrez des messages un peu plus explicites en provenance de l'application.

1.3. Help

Aucune aide pour l'instant. La seule documentation est celle que vous êtes en train de lire.

1.4. The run button

Utilisez le bouton (RUN) pour exécuter le script courant (celui qui apparaît dans l'éditeur).

2. The editor panel

Le panneau d'édition est le panneau principal de l'application. Il contient :

- Une barre de titre avec le nom du fichier
- Une zone de texte basée sur le package JEditSyntax (utilisant la coloration syntaxique pour une meilleure lisibilité de vos scripts).
- Une barre d'état extrêmement rustique indiquant le numéro de la ligne courante (afin d'aider au débuggage des scripts)

3. La ligne de commandes

Le panneau incluant la ligne de commande contient un champ pour saisir la commande, et deux boutons.

Le champ de saisie permet de saisir et d'exécuter des scripts très courts comme :

```
print(Math.sqrt(3*3 + 4*4));
```

ou

```
for(i=1;i++<16;) {sb=new StringBuffer(); for (j=0;j++<16;)
      {sb.append((char)(16*i+j)).append(' ');} print(sb);}
```

--> imprime la table des caractères locale dans le panneau de sortie.

Pour exécuter une telle commande, vous pouvez soit appuyer sur `la touche Entrée` du clavier après le point virgule, soit appuyer sur le bouton Run situé à droite de la ligne de commande.

Vous pouvez revenir aux commandes précédemment exécutées en vous servant des flèches du clavier.

Le dernier bouton de la ligne, Clear output, concerne le panneau de sortie qui concerne les sorties en provenance de tous les types de scripts.

4. Le gestionnaire de scripts

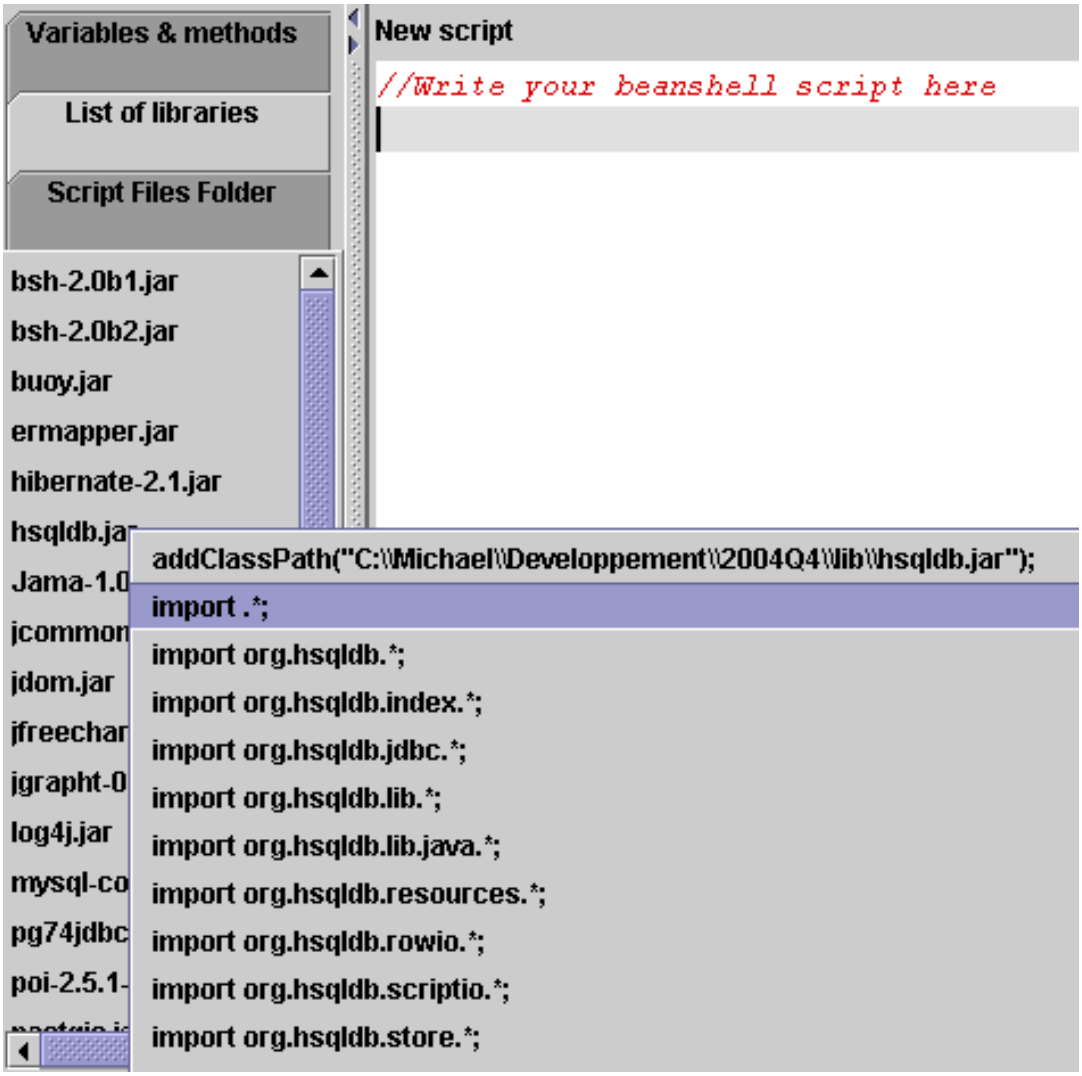
Le gestionnaire de scripts est un composant lui-même composé de trois panneaux :

4.1. Le répertoire de Scripts

C'est un sélectionneur de fichier dont la racine est un répertoire défini dans le menu options (et sauvegardé dans le fichier `BeanShellEditor.properties` file. Ce n'est rien d'autre qu'un moyen d'accès facile à vos scripts favoris.

4.2. La liste de bibliothèques

Ce panneau contient une liste des bibliothèques (fichiers jar) contenues dans le répertoire défini à cet effet grâce au menu options. Le nom de ce répertoire est également sauvegardé dans le fichier `BeanShellEditor.properties`.



Un click droit sur le nom d'une bibliothèque vous permet d'ajouter l'instruction addClassPath() qui convient à votre script (la ligne est insérée au niveau du curseur). Le menu contextuel vous propose également les packages contenus dans la bibliothèque et vous pouvez procéder à l'import des packages que vous voulez utiliser de la même façon.

4.3. Variables et methodes

Liste des variables et méthodes disponibles en ligne de commandes. Elle inclut les variables et méthodes définie dans l'application BeanShellEditor elle-même comme `bsh` et `bshEditor` ainsi que les variables et méthodes définies lors de l'exécution du dernier script. Elle n'inclut pas les commande propres à `BeanShell` comme la commande `print()` (pour ces méthodes, reportez vous à la documentation BeanShell).

Si vous cliquez sur un nom de variable ou de méthode, celle-ci sera automatiquement ajoutée à votre ligne de commande.

5. Le panneau de sortie

C'est un panneau texte sur lequel sont dirigés messages d'erreurs et instructions print(). Vous pouvez supprimer le texte du panneau en utilisant le bouton clear et vous pouvez l'éditer afin d'en modifier le contenu.

Avertissement

Si vous lancez un processus de longue durée, les sorties risquent d'être bloquées tout le temps du processus. Pour éviter cela, vous pouvez inclure votre processus dans un nouveau Thread, comme dans cet exemple parfaitement inutile :

```
class LongProcess extends Thread {
    public void run() {
        for(i=0 ; i<1000000 ; i++) {
            // process element i
            if(i%1000==0)print("" + i + " elements processed);
        }
    }
}
new LongProcess().start();
```

Chapitre 3. Et maintenant, écrivons quelques scripts

1. Scripts courts en ligne de commande

Vous pouvez évaluer des expressions que vous n'avez pas besoin de garder.

Par défaut, l'interpréteur BeanShell importe les package java suivants :

- java.lang
- java.io
- java.util
- java.net
- java.awt
- java.awt.event
- java.swing
- java.swing.event

Si vous utilisez une classe d'un autre package, vous devez utiliser son nom complet :

```
print(new java.text.SimpleDateFormat("EEEE d MMMM yyyy HH:mm:ss")
     .format(new Date()));
```

Pour en savoir plus sur la syntaxe BeanShell, les commandes BeanShell et toutes les possibilités de ce fantastique outil, téléchargez sa documentation à www.beanshell.org [http://www.beanshell.org].

2. Définir des variables, des méthodes et des classes

Dans l'éditeur BeanShell, vous pouvez définir des variables, des méthodes et des classes comme dans le langage java. Une fois qu'une variable ou qu'une méthode a été initialisée, vous pouvez l'utiliser plus loin dans le script ou simplement l'appeler à partir de la ligne de commande ou encore à partir d'un autre script.

2.1. Définir des variables

```
name = "Michaël";
dir = "D:/DATA/MYBOOKS/";
```

2.2. Ecrire une nouvelle methode

Une nouvelle méthode peut ressembler à cela :

```
dist(double x1, double y1, double x2, double y2) {
    dist = Math.sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
    print("Distance from ("+x1+","+y1+") to ("+x2+","+y2+") = " + dist);
    return dist;
}
```

Vous pouvez utiliser cette méthode définie dans le script en y ajoutant la ligne :

```
dist(0,0,1,1);
```

et exécuter la ligne à l'aide du bouton RUN.

Vous pouvez également exécuter le script sans aucun appel à la méthode que vous venez de définir, et appeler celle-là à partir de la ligne de commandes.

2.3. Implementer une interface

Vous pouvez implémenter une interface comme dans l'exemple suivant :

```
panel = new JPanel();
frame(panel);
background = new Color(127*256*256);

panel.addMouseWheelListener(new MouseWheelListener(){
    public void mouseWheelMoved(MouseWheelEvent e) {
        background = new Color(background.getRGB() + e.getWheelRotation()*1000)
        panel.setBackground(background);
    }
});
```

2.4. Créer une nouvelle classe

Avec BeanShell, vous pouvez créer de nouvelles classes d'objets comme si vous étiez en pur java :

```
public class Country {
    String name;
    int population;
    String capital;
    public Country(String name, int population, String capital) {
        this.name = name;
        this.population = population;
        this.capital = capital;
    }
    public String toString() {
        return name + "\n\tpopulation : " + population + "\n\tcapiatl : " + cap
    }
}
// Now you can call the class constructor...
print(new Country("France",60000000,"Paris").toString());
```

3. Débuggage de scripts

Sous la zone d'édition, vous pouvez voir le numéro de la ligne où se trouve le curseur. Celà vous aidera à localiser une erreur envoyée par l'interpréteur BeanShell qui précise le numéro de la ligne à laquelle l'erreur s'est produite :

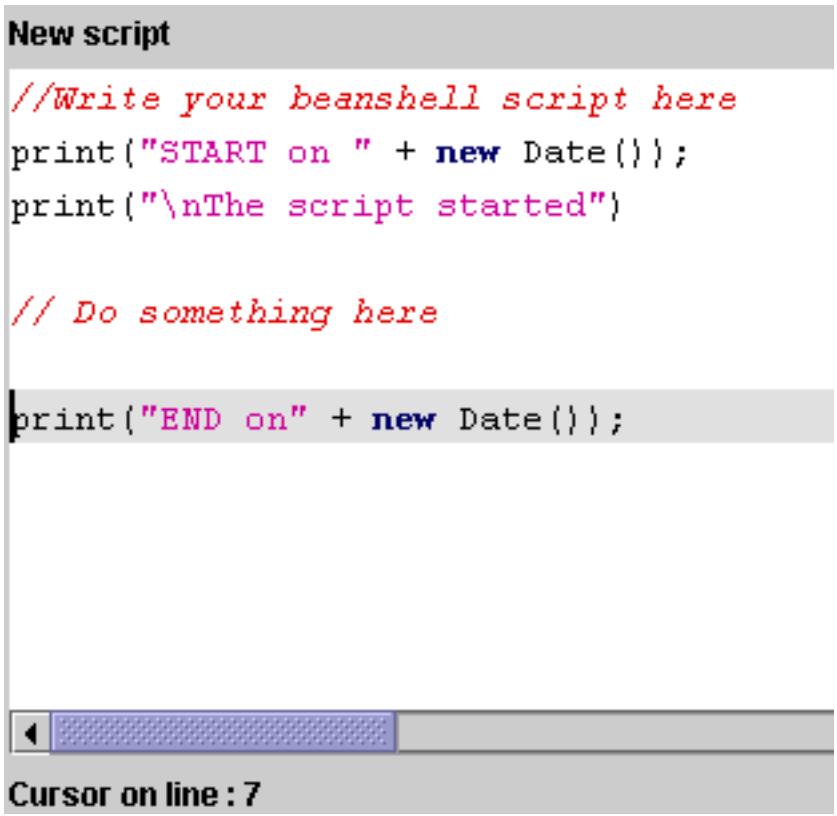
```
//Write your beanshell script here
print("START on " + new Date());
print("\nThe script started")

// Do something here

print("END on" + new Date());
```

Si vous exécutez ce script, vous obtiendrez le message suivant :

```
START on Wed Nov 10 11:13:51 CET 2004
```

```
Parse error at line 7, column 1.  Encountered: print
at bsh.Parser.generateParseException(Unknown Source)
at bsh.Parser.jj_consume_token(Unknown Source)
...
```

L'interpréteur a pu exécuter la première instruction mais le parser a généré une erreur en essayant de lire la suite. Il annonce une erreur en ligne 7.



En fait, le problème se tient sur la ligne précédent la ligne 7 (commentaires exclus), à savoir la ligne 2 à laquelle manque le ";" final de rigueur.

Ajoutez un point virgule à la fin de la ligne 2 et relancez le script.

4. Utiliser le gestionnaire de scripts

5. Utiliser le paneau liste de bibliothèques

6. Utiliser la liste des variables et méthodes

Annexe A. Version history

- **Version 0.1.1** (20 nov 2004)
  - Internationalisation : Pour ajouter un fichier de traduction, créer un fichier BeanShellEditor_i18n_ll_CC.properties (où ll est le langage et CC le pays), et placer le à la racine du fichier jar. Utiliser l'un des fichiers existants comme modèle.
- **Version 0.1.0** (13 nov 2004)
  - Première version publique.

Annexe B. TODO

- Completer cette documentation
- Inclure cette documentation dans l'aide en ligne de l'application.