BeanShellEditor

Version 0.1.1

Michaël MICHAUD

BeanShellEditor: Version 0.1.1

Michaël MICHAUD

BeanShellEditor is a script editor for BeanShell [http://www.beanshell.org/home.html]. It may be used as a standalone application or as a plugin in another application. In both cases, one can write short programs very easily, access a script file in one click (through the script manager), import new packages easily... Thanks to the famous BeanShell scripting language, BeanShellEditor put the power of java in the hand of non programmers. The UI uses the excellent Buoy library, and the editor uses the JEditSyntax package, which is an old version of the famous JEdit text editor.

This program has the following dependancies :

- BeanShell library from Pat Niemeyer (library is LGPL)
- Buoy from Peter Eastman (library is in the public domain)
- JEditSyntax (MIT license)

Table of Contents

Chapter 1. Introduction

1. What is BeanShellEditor

BeanShell Editor is a script editor for BeanShell. BeanShell is a small, free, embeddable, Java source interpreter with object scripting language features, written in java. Its mascott is  .The author of this library is Pat Niemeyer. You will find the binary, the source, and the docucumentation on beanshell site [http://www.beanshell.org]. BeanShellEditor is just an application making the use of beanshell more easy (you'll find also JConsole in BeanShell distribution an application having the same goal as BeanShellEditor, but which did not match my specific needs).

2. How to install BeanShellEditor

To install BeanShellEditor, you need :

- a computer with a jvm 1.4+ installed (downloadable at java.sun.com [http://java.sun.com])
- the bsheditor.jar file, which contains the the application (includes JEditSyntax package)
- the beanshell library (also downloadable at www.beanshell.org [http://www.beanshell.org])
- the buoy library (also downloadable at buoy.sourceforge.net [http://buoy.sourceforge.net])

The distribution does not include all the libraries. You may have to download beanshell and buoy on the net (these are small libraries, under 1 MB)

2.1. Install BeanShellEditor as a standalone application

If `BSHEDITOR` is your installation directory, you'll need to have :

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

If the `BeanShellEditor.bat` is not present in your distribution, you just have to edit a file with the following line :

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

or

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

if you want to hide the dos console after the application has been launched.

To launch the application, double-click on the BeanShellEditor.bat file.

Alternatively, on modern machines, you can put the 3 jar files in a directory and double-click on the bsheditor.jar

Any experience of running the programm under a linux box or a mac is welcome.

2.2. Install BeanShellEditor as a PlugIn

Installing BeanShellEditor as a PlugIn is very simple :

Put `bsheditor.jar`, `beanshell.jar` and `buoy.jar` in your classpath.

Create a button or a menu in your application which creates a new BeanShellEditor instance :

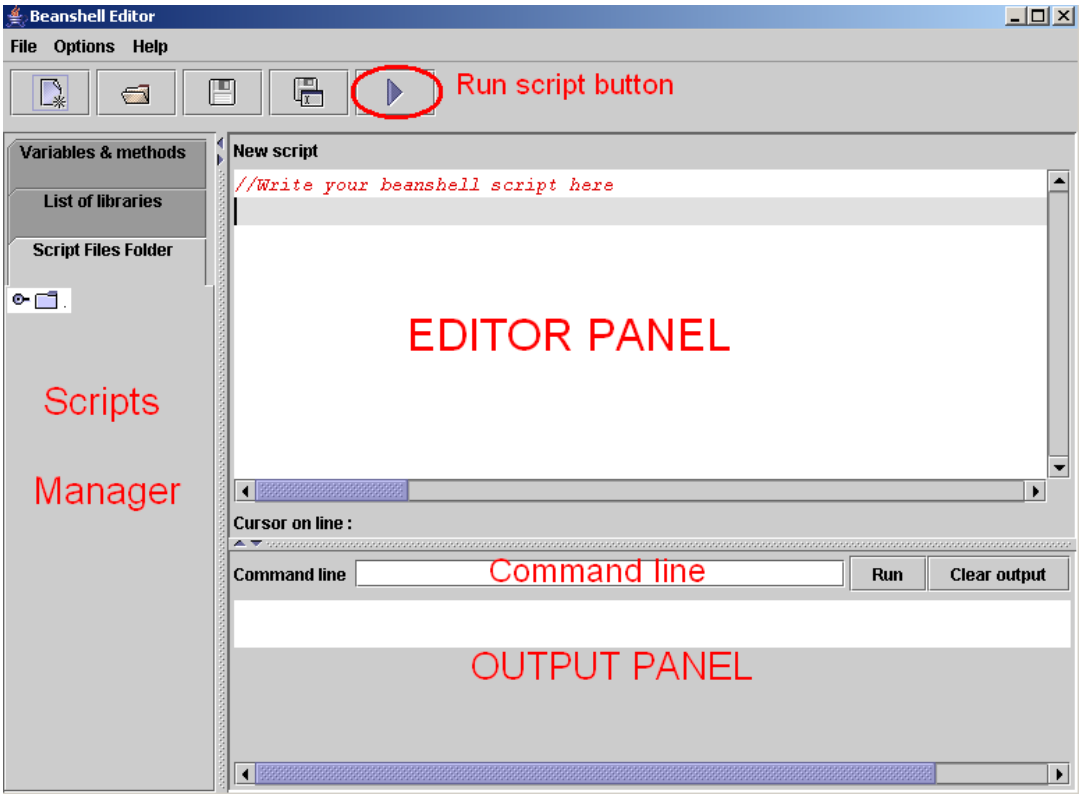
You must use the constructor taking a map argument. The map will contain variable names as keys and your application main objects as value..

For example (method called by your application to create the 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);
}
```

Chapter 2. BeanShellEditor main features

Here is a screen shot of the BeanShellEditor.



1. The menu and the toolbar

Here are the commands you will find in the menu / toolbar

1.1. File menu

The file menu item and the toolbar include :

- New file : to edit a new script file. You can also use the button in the tool bar
- Open file : to open an existing file. You can also use the button in the tool bar or the script manager (see section script manager)
- Save file : to save the current file. You can also use the button in the toolbar
- Save as : to save the current file with another name. You can also use the button

1.2. Options menu

BeanShellEditor options include :

- Choose scripts folder : to choose the folder where you want to put your scripts. Bean-

ShellEditor represents this folder as a tree in the script manager from where you can acces your previous scripts with one click.

- Choose jars folder : you can choose a folder containing the java libraries you want to use in your shells. In the Script manager, you can select the Jars folder pane which lists the jar files in your jar directory. A right click on a library name let you include the library in the classpath (`addClassPath`) of your current script or import new packages (`import`).
- Choose start file : to choose the starting script file, a BeanShell script which may always be run (see also below) before the edited script execution. The starting script is not run again before you run a command line.
- Always execute start file : if this menu item is marked, the starting script will always be executed before a script is run from the editor panel.
- Verbose outputs : if this menu item is marked, you'll get some more message from the BeanShellEditor application

1.3. Help

Nothing yet. The only documentation is the one you are reading now.

1.4. The run button

Use  to run the script appearing in the editor.

2. The editor panel

The editor panel is the main part of the application. It contains :

- A title bar with the file name
- A text area based on JEdit syntax package (making your scripts more readable thanks to the syntax colouring).
- A very rough state bar giving the line number where your cursor is (to help dubugging your scripts)

3. The command line

The command line panel contains one text field and two buttons.

The text field let you run very short scripts as :

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

or

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

(print a table of local characters in the output text area).

To run the one line script you may either input 'Enter' or press the run button after the text field area.

You can navigate through previously executed commands with the up/down keys.

The last button of the line, Clear output, concerns the output panel which is a common output area for both script editor and command line outputs.

4. The scripts manager

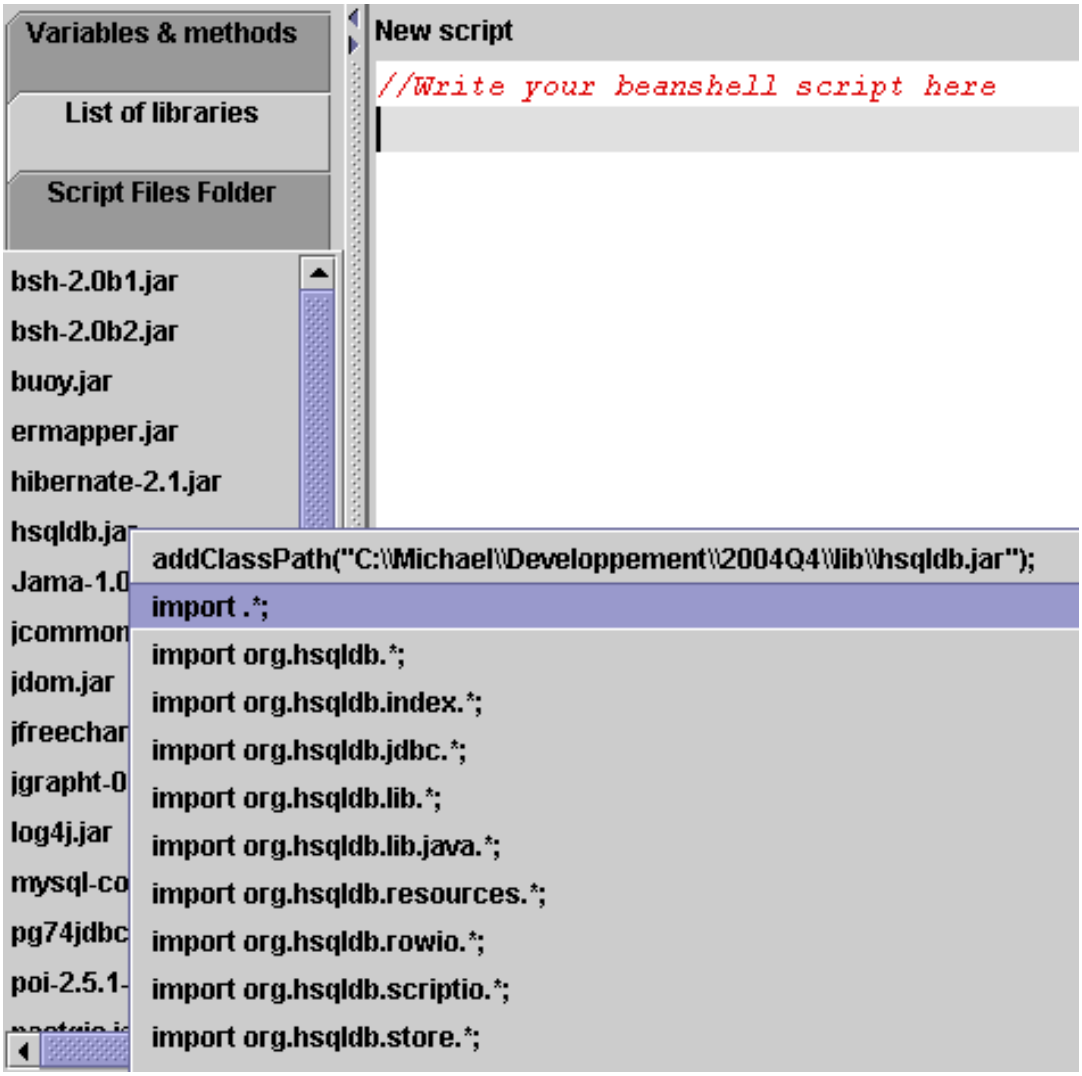
The script manager is a tabbed pane with three panels :

4.1. The Script Files Folder

This is a file chooser which root is the script folder defined through the options menu (and written in the `BeanShellEditor.properties` file). It is nothing but an easy way to open your favorite scripts and to run them.

4.2. The List of libraries

This panel contains a list of libraries (jar files) contained in a user defined folder (the folder is defined using the options menu and written in the `BeanShellEditor.properties` file).



With a right click, you can add an `addClasPath()` statement or an `import` statement in your current sript (you must place your cursor at the start of the line where the statement has to be inserted.

4.3. Variables and methods

List of variables and methods available in the command line. It includes variables and methods defined in the BeanShellEditor (`bsh` and `bshEditor`) and variables and methods defined in the last executed script file. It does not include beanshell commands like `print()` (see beanshell documentation).

If you click on a variable or on a method name, this one is included in the command line.

5. The output panel

The panel displaying your script outputs. This is a simple text area that you can clear using the clear button and that you can modify as any editable text area.

Warning

If you launch a long process, the outputs will be blocked until the main process is finished. To avoid that, you can include your process in a thread like in this totally unuseful example :

```
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();
```

Chapter 3. And now, let's write some scripts

1. One line scripts with the command line

You can use the command line to evaluate expressions you don't want to save.

By default, the beanshell interpreter imports the following packages :

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

If you want to use classes from other packages, you can use their full name :

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

To know more about beanshell syntax, beanshell commands and all the capabilities of this fantastic tool, download the documentation at www.beanshell.org [http://www.beanshell.org].

2. Setting variables, methods and classes

In the beanshell editor, you can set variables, methods and classes, as in the java language. After a variable or method has been defined, you can use it further in the script, or just call them from the command line or from another script.

2.1. Setting variables

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

2.2. Writing a method

A method may just look like :

```
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;
}
```

You can use this method in the script, adding the following line

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

and running it from the upper run button.

You can also run the script without any call to the method you have defined, and call the method from the command line.

2.3. Implementing interface

You can implement an interface :

```
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. Creating a new class

With beanshell, you can create new classes as if you were coding in pure 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. Debugging scripts

Below the editor, you'll find the current line number to help you debugging when beanshell send you an exception :

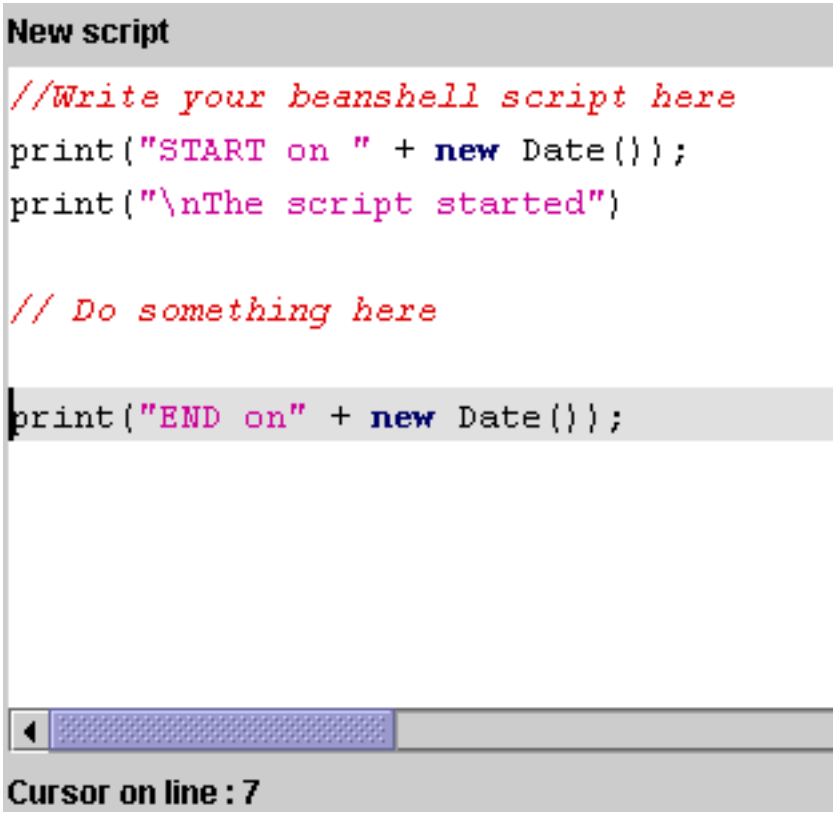
```
//Write your beanshell script here
print("START on " + new Date());
print("\nThe script started")

// Do something here

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

If you write such a script, you'll get the following message :

```
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)
...
```

BeanShell interpreter successed in running the first printing instruction but the parser failed trying to run the second. The debugger found a problem at line 7.



In fact, the problem lies here in the previous uncommented line which miss a ";".

Add a ; at the end of third line, and run again your script file.

4. Using the script files folder

5. Using the list of libraries

6. Using the list of variables and methods

Appendix A. Version history

- **Version 0.1.1** (20 nov 2004)
  - **Internationalization** : to add a language, write a BeanShellEditor_i18n_ll_CC.properties file where ll is the language and CC the country, and put it at the root of the jar. Use one of the existing file as an example.
- **Version 0.1.0** (13 nov 2004)
  - First public release

Appendix B. TODO

- Complete this documentation
- Include this documentation in the application help