JIF a java editor for Inform

Alessandro Schillaci - Peter F. Piggott

February 23, 2013

Overview

JIF is an Integrated Development Environment (IDE), written entirely in Java, for the creation of text adventures based on Graham Nelson's Inform standard. With JIF it's possible to edit,compile and run a text adventure in the z-code and glulx formats.

Inform 7 has been developed with a new Integrated Editor but JIF can be used in any Inform 6 or 7 Environments

The project has started in the 2003 and now it's in a stable state for production, and the latest version is the 3.6 (Feb 2013). This project is released under the GNU GPL Licence.

Main Features:

- Written in Java (multiplatform, it runs on any Operative System: Windows, Linux, Unix, MacOSX)
- Syntax Highlighting support
- Inform and Glulx support
- Code Assistant and Template Manager, to speed your code
- Multi-language support: English, Italian, Spanish, German and French
- Project management: support for compilation of wide projects
- Switches Manager: customize your switches passed to compiler.

Configuration

What's new?

Jif has been improved, many bugs have been fixed and the configuration management revised. The main change is that now Jif configuration information is stored into a single configurations file. You can edit this file ("Jif.cfg") out of Jif and restart Jif to see your changes take effect. You can also edit this file within Jif (without the need to re-start Jif): just go to the Options-> Jif INI file (menu) and edit it. Then save and see the changes on Jif.

For the Linux/Unix/MacOsX users: now Jif can load the configuration from a different file. Just pass a parameter to the VM to tell Jif which file it has to use as configuration file.

E.G. : launch Jif with these parameters
java -Duser.language=en -Duser.region=US
-Djif.configuration=[NEWPATHFILE] -cp . -jar Jif.jar

- The flags **-Duser.language=en** and **-Duser.region=US** let Jif to load the English version
- Tha flag **-Djif.configuration** lets Jif to use [NEWPATHFILE] as configuration file (instead of **JIF.cfg**)

From this version, you can put the Jif.jar file where you want to and the configuration file in another place.

So, Let's see how to change/customize your configuration file for the Jif. Open the "Jif.cfg" file with Jif (or another text editor) and follow these sections.

Notes:

- If you edit the configuration file within Jif, you can save it and see the changes take effect, without restarting Jif. If you want to edit the config file within another text editor, you can do that, but you have to restart Jif to see your changes.
- If you want to edit the Jif.cfg file to make your own copy, please create a backup first of the original file. If you miss something, you wouldn't be able to run Jif properly. In this case, don't send me any feedback error on what you've done. Just rename your Jif.cfg backup and retry. Keep this in mind and keep on reading this doc.

[ALTKEYS] Section

Jif is highly customizable.

If you want to execute a command (an external command) just by pressing a key combination you can do it. If you want to type "a"+"ALT" and see the "^" character or something like this, read this section.

Example:
[ALTKEYS]w,{}
[ALTKEYS]a,^
[ALTKEYS]t,~
[ALTKEYS]s,‘
[ALTKEYS]q,[]
[ALTKEYS]p,|
[EXECUTE]e,explorer.exe
[EXECUTE]d,hh.exe C:\Jif\doc\prova.chm ...

Ok, this should be self explained. You can add a key "listener" to Jif, simply add a line like these. Just follow this format:

[ALTKEYS]<CHARACTER +"ALT">,<CHARACTER/STRING TYPED>

or

[EXECUTE]<CHARACTER + "ALT">,<COMMAND TO EXECUTE>

Let's go for an example. You want to add a key to type the "()" (open-close parenthesis) in just one key press? You want to associate it to the "k" key? Simply, add this line to your Jif.cfg file.

[ALTKEYS]k,()

Open Option->Jif INI file menu, edit the file adding the previous line and save it. Now your changes are ready. Open a file (or create a new one) and press "ALT+k". You should see the "()" typed at the current cursor position.

Another example? Do you want to launch a pdf reader with the DM4 manual within Jif? Simply add this line to the [ALTKEYS] Section.

[EXECUTE]o,C:\Acrobat 7.0\AcroRd32.exe C:\DM4.pdf

Notes:

- on Linux/Unix Systems this could have some problems. Just test it and feedback to me. On win32 systems this runs fine.
- pay attention not to use ALT keys combinations that are already used by the operating system (like ALT+X and others). In this case this "re-mapping" of ALT keys will be unuseful. (and if you type ALT+X, Jif will shutdown in many operating systems). So, be careful and use your mind

[HELPEDCODE] Section

Ok. What's the meaning of "helpedcode"? Sorry, this isn't code completion. This is abbreviation management.

You want to insert the **"animante"** keyword just typing **"an"** characters? You can do with the HELPEDCODE section.

[HELPEDCODE]an,animate

Now you can insert "an" and press the CTRL+SPACE keys. Jif will substitute the "an" characters with the "animate" keyword.

But, hey, stop. Give up. Wait a moment. Rewind the tape. What the hell is the difference between the ALTKEY SECTION and the HELPEDCODE section?

Ok. You're right.

In the HELPEDCODE section you can choose where the cursor will be after the substitution. You can also simulate a RETURN key press or a TAB key press. Consider this example.

[HELPEDCODE]ba,before [;@],[ret]after [;],

You insert "ba" and press the CTRL+SPACE. Jif will insert: before[;(THE CURSOR IS HERE)],(<—RETURN)

after[;],

Note:

[ret] = RETURN character
[tab] = TAB character
@ = Cursor Position

So take a look at the HELPEDCODE section and you will find that this is simpler than you think.

[HELPEDCODE](,(@)
[HELPEDCODE][,[@]
[HELPEDCODE][;[;@],
[HELPEDCODE]ab,absent
[HELPEDCODE]af,after [;@],
[HELPEDCODE]an,animate
[HELPEDCODE]ba,before [;@],[ret]after [;]

[MAPPING] Section

So, what about the ZCODE characters? I mean the "@‘e" that stands for è and so on. If you want to write a game which is compatible with most of the Inform Interpreters (zcode), you should use these special characters. Jif can manage this for you.

Let’s take a look at the MAPPING section.

[MAPPING]é,@’e
[MAPPING]è,@‘e
[MAPPING]à,@‘a
[MAPPING]ì,@‘i

Nothing hard here. It is all linear. If you press the "è" character, for example, Jif will type the ZCODE special characters "@’e" into the main textarea, at the current cursor position.

[MENU] Section

Ok. Prepare yourself: things are a little bit harder in this section.

When the cursor is in the main textarea (the input code area) and you press the mouse’s right button, a popup menu will appear. The main Menu Item is "Insert new". This menu item lets you insert a piece of text. You can customize the menu entries in this menu. Let’s look at the actual configuration entries.

[MENU][Class]*
[MENU][Class]Generic Class,Class CLASS_NAME §has ATTRIBUTE;
[MENU][Class]Room Class,Class Room§ has light; ...

The first line is to define a new Menu: you’ll see "Class" as a menu name. Ok. Ok. ok. And what about the "Class" menu’s contents? Yes. See the last two lines. "Generic Class" and "Room Class" are sub-items for the main

"Class" menu. If you click on "Room", Jif will append to your current file, this string:

Class CLASS_NAME
has ATTRIBUTE;

So, the "insert new" menu structure will be:

- Insert new
  - Class
    - Generic Class
    - Room Class

Note that the "§" character is converted into an enter command (Just as if the user had pressed the "Enter" key). Do you want to see an example in action? Yes, I guess, you want to.

The right format for the definition of a new menu is:

[MENU][<Main Menu Name>]*

The MenuItems are added to a Main Menu in this way:

[MENU][<Main Menu Name>]<Menu item name>,<piece of text you want to insert>

Let's create a new menu: "NPC" and create a new menu item. Open the "Jif.cfg" file. Go to the MENU Section and append this:

[MENU][NPC]* [MENU][NPC]Nick,NPC nick "Nick"§ with name 'nick',§ before [;],§ after[;],§ has male proper;

Ok. Finished. Go to the current open TextArea and click on the right mouse button. Choose: "Insert new"->"NPC"->"Nick" Jif will append this code:

Nick,NPC nick "Nick"
with name 'nick',
before [;],
after[;],
has male proper;

That's all. If you create an interesting popup menu configuration, please send it to me: I'll add it into Jif official site. Thanks.

[SWITCH] Section

If you're about to compile an inform game, you should feel confident with the inform "switches". These are just parameters passed to the compiler and are used to tell the compiler how create to create your zcode (ulx) file. Of course, with Jif, you can manage them without any effort.

In the "Jif.cfg" file there is a list of switches available within Jif. Let's look at the [SWITCH] section.

[SWITCH]-c,off

[SWITCH]-d,on
[SWITCH]-d2,off
[SWITCH]-f,off

Ok. This part isn't the hardest one. Every switch can be set to "on" or "off". If it is set to "on", Jif will use it and will pass it to the compiler. If you need a new switch that doesn't exist in the Jif.cfg file, you can add it to the SWITCH section and save the configuation file. Within the Switch Manager you can control which switches you wish to pass to the compiler. Open Switch Manager to see the switch you've added.

Note: when you create/save a project, all the switches states are stored into the projects jpf file (which stands for [J]if [P]roject [F]ile). So you can have a default configuration for the switches, and a different one for each project you create. Just create a new project. Whenever you save it, Jif will save the current switches configuration also.

[SYNTAX] Section

Jif supports an Inform syntax highlighting system: all the keywords have different colours. This information is stored in the SYNTAX section. Let's look at the config file again.

[SYNTAX][attribute]absent
[SYNTAX][keyword]creature
[SYNTAX][property]add_to_scope
[SYNTAX][verb]Answer

As you can see, Jif supports 4 different keywords types and you can choose 4 different colours for attribute, keyword, property and verb (within the Jif Settings Dialog). Just add the keywords you would like highlighted into the "Jif.cfg" file. Save the file and you will see that now Jif recognizes and highlights the new keywords with the type colour listed.

Note: These keywords are not case sensitive (because Inform's syntax is generally not case sensitive, as you probably know). So you can write the keywords in any case: absent or Absent or AbSeNt or ABSENT

Jif will treat them all the same and colour them as an attribute. But you only need to add one entry in the Jif.cfg file:

[SYNTAX][attribute]absent

[SYMBOLS] Section

This section is like the [ALTKEYS] one, but that Jif will show a Symbols List to choose from. You can add anything you like, maybe something you don't use frequently in your inform code or just keys that don't exist in you keyboard layout like @<<,@>>,ä,à,á,â,ã,å and so on.

The format is the same:

[SYMBOLS]@<<
[SYMBOLS]@>>
[SYMBOLS]ä
[SYMBOLS]à
[SYMBOLS]á
[SYMBOLS]â
[SYMBOLS]ã
[SYMBOLS]å

You can add a new "symbol", save and click on the Symbols list Button (on the toolbar) to see the list with the new entry. Use it whatever you like.

[PATH] Section

This is the Jif's core configuration. If you don't set the inform compiler path, Jif will not be able to launch the compilation process or execute your adventure within the inform / glulx interpreter.

The first time you run Jif, this section will be empty. You can fill it in two ways:

1. go to the Options->Settings->Path panel and fill them. Then Save.
2. open the Jif.cfg file and edit them by hand (suggested only if you know what you're doing.)

When you've set and saved the Jif.cfg file, you'll have a situation like this:

[LIBRAYPATH]C:\Inform\Lib\Base
[LIBRAYPATHSECONDARY1]C:\Inform\Lib\Contrib
[LIBRAYPATHSECONDARY2]
[LIBRAYPATHSECONDARY3]
[COMPILEDPATH]C:\Inform\Games
[INTERPRETERZCODEPATH]C:\Inform\Bin\interpreter\Frotz\Frotz.exe
[INTERPRETERGLULXPATH]C:\Inform\Bin\interpreter\Gargoyle\Gargoyle.exe

[COMPILERPATH]C:\Inform\Bin\compiler\inform.exe
[BRESPATH]C:\Inform\Bin\tools\Blorb\bres.exe
[BLCPATH]C:\Inform\Bin\tools\Blorb\blc.exe

These are my current settings, yours may well be different. So, I'll explain them.

[LIBRAYPATH]
[LIBRAYPATHSECONDARY1]
[LIBRAYPATHSECONDARY2]
[LIBRAYPATHSECONDARY3]

These are the Libraries paths. You can set the main library path. Then you can manage 3 different Library locations. In case you have source file in different packages, and you don't want to merge all in one directory.

[COMPILEDPATH]

This is the path where Jif will put the compiled files. If you use the project management (this is recommended) the compiled files will be put in the project main directory. Be sure you've set this path in ANY CASE.

[INTERPRETERZCODEPATH]
[INTERPRETERGLULXPATH]

Do you use Frotz as Inform Interpreter? Ok. Set the [INTERPRETERZ-CODEPATH] path, to the absolute path to Frotz. Same thing for the Glulx Interpreter (Wingit, WinGlulxe, Gargoyle). Here you can choose your favourite Interpreter. Just do it.

[COMPILERPATH]

Yes, this is the most important path to set in the Jif.cfg file. The Inform compiler. You know what it is, don't you? If you don't, please take time to read about it and understand what it is and how to use it. Read the DM4, the Inform Beginners Guide to Inform an Inform tutorial.

Don't use Jif until you understand the basic idea of what a "z-code" file is.

[BRESPATH]
[BLCPATH]

These tools aren't mandatory. These are for the blb format. If you only want to create a zcode adventure, you can leave them blank. If you want to create a BLB file (or ULX) within Jif, you have to set them point to the right absolute paths for these two tools. Then you can make the resources, compile to ulx format and (finally) create a single-bundle BLB file to distribute to the waiting world, out there.

[SETTINGS] Section

This section is created by JIF. Do not touch it, if you don't know how to manage it. In this section Jif stores all the configuration for the graphic layout, such as the colour management, the actual options checked and so on.

Warning!!! Do not edit this section by hand. Use the configuration Panel.

[RECENTFILES] Section

This section is created by JIF. Do not touch it, if you don't know how to manage it. This section stores the list of last opened files.

Warning!!! Do not edit this section by hand. Use the configuration Panel.

Shortcuts Section

Edit Mode

- CTRL+N New file
- CTRL+O Open file
- CTRL+S Save file
- CTRL+SHIFT+S Save all
- CTRL+Q Close file
- CTRL+P Print
- ALT+X Exit
- CTRL+A Select all
- CTRL+X,SHIFT+CANC Cut
- CTRL+C,CTRL+INS Copy
- CTRL+V,SHIFT+INS Paste
- CTRL+Z Undo
- CTRL+Y Redo
- TAB (or ALT + ->) Right shift selected text block
- SHIFT+TAB (or ALT + <-) Left shift selected text block
- CTRL+, Comment selection
- CTRL+. Uncomment selection
- CTRL+F2 Set/Reset Bookmark
- F2 Go to next bookmark
- F11 Toggle FullScreen
- F1 View help

Search Mode

- F3 Search String (from the current file)
- CTRL+J Jump to the current word's definition
- CTRL+F3 Search String (from the whole project)
- CTRL+F Find selected string
- CTRL+R Find and replace
- CTRL+V Find object in source

Project Management

- CTRL+F4 New project
- F4 Open project

Inform

- F9 Compile
- CTRL+F9 Run
- F12 Build all
- CTRL+F12 Make blb
- Right Button (mouse) Menu Popup 1 (from the Text Editor) Menu Popup 2 (from the Project file list)

Install JIF and other components

From the 3.5 version of JIF (and above), the main application is released in the JAR format. To install it, simply double-click on the JIF.jar file.

Since the 3.6 version, a Java Runtime 1.7 version is required.

The same author has released an "Inform Pack": a collections of Inform/Glulx tools pre-configured to work. (see http://www.slade.altervista.org)

Copyright and License

Credits

A special thank goes to Peter F. Piggott for his constant help to improve JIF.

JIF Development:

- Alessandro Schillaci
- Luis Fernandez
- Peter F. Piggott

Contributors:

- Paolo Lucchesi
- Vincenzo Scarpa
- Baltasar G. Perez-Schofield
- Christof Menear
- Giles Boutel
- Javier San Josè
- David Moreno
- Eric Forgeot
- Max Kalus
- Adrien Saurat
- Alex V Flinsch
- Daryl McCullough
- Giancarlo Niccolai
- Ignazio di Napoli
- Joerg Rosenbauer
- Matteo De Simone
- Tommaso Caldarola
- Jenesis

Third part components:

- Everaldo: Some Icons used by Jif (www.everaldo.com)

Contacts

Author Site: http://www.slade.altervista.org/
Main Project Site: http://developer.berlios.de/projects/jif/
Email: silver.slade@tiscalinet.it (Alessandro Schillaci)