Speare Code Editor
The Small Ruby, mruby and Rails IDE for Ruby Development

Version 0.0.2
Update: 27 Mar 2021

Free, Lightweight, Open Source, Extendable Flexibility

Speare (http://www.sevenuc.com/en/speare.html) is an ultra lightweight code editor and a small IDE that has an efficient code navigation and call routines tracing ability, and has integrated debugging environment for C, C++, Ruby, mruby and Ruby on Rails. It was originally developed to providing a native scripting language debugging environment that seamlessly integrated with C and C++. It not only has very concise user interface but also has a very flexible architecture to easily add a new programming language in it.

For general information of debugging C and C++ extension for mruby/c, mruby and Ruby with Speare code editor, please refer this document: http://www.sevenuc.com/download/Small_IDE_for_C_and_C++_development.pdf

Debug Ruby Projects

1. Show the debug toolbar

Click main menu "Navigate" → "Toggle Output".

2. Debug toolbar

From left to right: Start, Stop, Step Into, Step Out, Run To/Continue, Step Over/Step Next, Show Watches. The command "Run To" tell the debugger run to meet a breakpoint or an exception occurred or the program exited. On the rightmost there are three other icons, they are, search items in the stackview, siding stackview, and clean debug output.

Search in the debug output

Click in the output area to let it got focus and use the shortcut key "Command + F" to do the searching.

3. Socket Port

You can set the socket communication port number both used by Debug Server and the editor. Open the Preferences of Speare and select the "Extra" panel and then input your number.

Note: Please remember to empty the port number when you switched to debugging with the default built-in programming languages with default port number.

4. Watches

Watch List: click debug toolbar "Show Watches" to manage watched variables and expressions, batches of watched items can be removed by multiple selection.

a. Whenever values of watched variables changed, debug session will pause at that point.
b. The values of evaluated expressions will be sent as a *"virtual stack node"* that paired the expression and its value together and representing as JSON key and value.

When watched variables and expressions display in stackview, their nodes normally has a different icon and text colour, and always placed on the top of stackview.

Add Watches:

a. Click debug toolbar "Show Watches".
b. Right click stackview and select menu item "Watch Variable".

Remove Watches:

a. Click debug toolbar "Show Watches".
b. Right click stackview and select menu item "Remove Watch".

5. Run Extra Commands

a. Click main menu "Debug" → "Send Debug Command".
b. Right click stackview and select menu item "Send Command".
When send an extra debug command, the input box will prompt in stackview, so please click stackview to let it got focus before select menu item.

6. Show Stackview Item Values

Right click stackview and select menu item "View Value AS", variables can be configured to display as "Hex", "Decimal", "Octal", "UTF-8" and "Unicode" sequences.

Caution:
Please ensure all source files have been dragged in the workspace before start a debug session, because macOS app can't be allowed to access files outside of its sandbox.

Startup Debugging Session

1. Launch a Terminal.app window or tab, and start Debug Server.
Note: *The content directly printed by the debugging program will be displayed in the terminal instead of in the debug "output view".*

2. Click start button on the "Debug toolbar" to start debugging session.

3. Continually click "Step Over" button several times on the "Debug toolbar" to ignore some initialising steps in debugging session until it reached the main entry point.

mruby Debugger

The mruby debugger of Speare code editor is a patched version of mruby (http://mruby.org) that support remote debugging mruby project, currently support mruby version 2.0.1, 2.1.0, 2.1.2 and 3.0.0.

1. Install mruby debugging server

Download one of the mruby remote debuggers:
http://sevenuc.com/download/mruby_debugger_2.0.1.tar.gz(362KB)
http://sevenuc.com/download/mruby_debugger_2.1.0.tar.gz(370KB)
http://sevenuc.com/download/mruby_debugger_2.1.2.tar.gz(368KB)
http://sevenuc.com/download/mruby_debugger_3.0.0.tar.gz(440KB)

Download mruby-2.0.1.tar.gz (518KB), mruby-2.1.0.tar.gz (585KB), mruby-2.1.2.tar.gz (663KB), or mruby-3.0.0.tar.gz (701KB) from https://github.com/mruby/mruby

```
$ cd mruby-2.x or  mruby-3.0.0
$ make
```
compile mruby and replace mrdb under bin directory with the corresponding version.

```
$ cd bin
$ ./mrdb   # start the mruby remote debugger.
```

2. Configuring Speare code editor

Launch Speare and open the Preferences of Speare and select the tab of "**Debug Settings**" then check on "**Enable mruby debugging**". Please remember to turn the option off when you switched to debug common Ruby applications.

3. Debug Session Start

Click "**Start**" button on the debug toolbar of Speare code editor.
Add breakpoint, step in, step out, step next, watch stack trace ...

Tips: Separate modules of your app with mruby gems instead of using require.

Ruby Debugger

The Ruby debugger of Speare code editor implemented as a client of rdebug-ide, and Ruby interpreter that has a rdebug-ide installed will be running as the debug server.

Ruby debugging environment support all kinds of Ruby interpreters, the version includes: 1.8.x, 1.9.x, 2.x, and JRuby.

Steps of start debugging session

1. Download and install debug gems

For Ruby 1.8.x: download: ruby-debug-base (0.10.4)
```
$ gem install --force --local ruby-debug-base-0.10.4.gem
$ gem install ruby-debug-ide
```

For Ruby 1.9.x: download: ruby-debug-base19 (0.11.25)
```
$ gem install --force --local ruby-debug-base19x-0.11.32.gem
$ gem install ruby-debug-ide
```

For Ruby 2.x:
```
$ gem install debase
$ gem install ruby-debug-ide
```

2. Start the debug server

$ rdebug-ide --host 127.0.0.1 --port 1234 --dispatcher-port 1234 -- main.rb
(Note: Please replace the main.rb file with your script file.)

For Ruby on Rails:

$ rdebug-ide --host 0.0.0.0 --port 1234 --dispatcher-port 1234 -- bin/rails s

3. Debug session start

Click "Start" button on the debug toolbar of Speare code editor.
Add breakpoint, step in, step out, step next, watch stack trace ...

4. Add condition breakpoint

Right click on the breakpoint, on the prompt menu, → select "Condition" and then input expression or use empty string to remove the condition, left click outside of the input box to close it and execute the command. e.g. x>5 means: Pause on the breakpoint only if x>5 is true.

5. Run Extra Commands

Right click in the stackview (bottom left side) and then input extra command when the debugging session paused. Left click anywhere outside of the input box to close it and the command will be directly send to the debug server.

a. Variables

. var const object: show constants of object.
. var instance object: show instance variables of object, object can be given by its id or an expression.
. var inspect: reference inspection results in order to save them from the GC.

b. Expression

. p expression: evaluate expression and print its value.
. pp expression: evaluate expression and print its value.
. eval expression: evaluate expression and print its value, alias for p.
. expression_info expression: returns parser-related information for the expression given 'incomplete'=true | false indicates whether expression is a complete ruby expression and can be evaluated without getting syntax errors.

c. Backtrace

. where: display frames.
. bt | backtrace: alias for where.
. up | down [count]: move to higher or lower frame.
. frame [frame-number]: Move the current frame to the specified frame number. (A negative number indicates position from the other end. So 'frame -1' moves to the oldest frame, and 'frame 0' moves to the newest frame.)

d. Jump
Change the next line of code to be executed.
. jump line: jump to line number (absolute).
. jump -line: jump back to line (relative).
. jump +line: jump ahead to line (relative).

e. Thread
. thread list: list all threads.
. thread current: show current thread.
. thread switch <nnn>: switch thread context to nnn.
. thread inspect <nnn>: switch thread context to nnn but don't resume any threads.
. thread resume <nnn>: resume thread nnn.
. thread stop <nnn>: stop thread nnn.

f. Type Set
. set_type <var> <type>: Change the type of <var> to <type>.

g. File Operation
. load file: read and parse file every time instead of require.
. file-filter on | off: enable or disable file filtering.
. include file | dir: adds file or dir to file filter (either remove already excluded or add as included).
. exclude file | dir: exclude file or dir from file filter (either remove already included or add as exclude).

Switch Ruby Interpreter
You can directly switch between any Ruby interpreter or your own version of Ruby and then config it to support rdebug-ide.

Appendix:
Make a fresh Ruby debugging environment

Step 1. Build an openssl library

```
$ download https://www.openssl.org/source/openssl-1.0.2t.tar.gz
$ tar -zxvf openssl-1.0.2t.tar.gz
$ cd openssl-1.0.2t
$ export KERNEL_BITS=64
$ ./config no-ssl2 no-ssl3 no-shared enable-ec_nistp_64_gcc_128 \
        --prefix=/Users/yeung/Public/Rdebug/openssl \
        --openssldir=/Users/yeung/Public/Rdebug/openssl
$ make && make install
```

Step 2. Build a ruby interpreter

```
$ download https://cache.ruby-lang.org/pub/ruby/2.1/ruby-2.1.2.tar.bz2
$ tar -jxvf ruby-2.1.2.tar.bz2
$ export LDFLAGS=-L/Users/yeung/Public/Rdebug/openssl/lib -lcrypto -lssl
$ export CFLAGS=-I/Users/yeung/Public/Rdebug/openssl/include
$ export PKG_CONFIG_PATH=/Users/yeung/Public/Rdebug/openssl/pkgconfig
$ cd ruby-2.1.2
$ ./configure --prefix=/Users/yeung/Public/Rdebug/2.x/ruby2 \
              --with-openssl-dir=/Users/yeung/Public/Rdebug/openssl
```

Alternative:
directly modify Makefile to add openssl library link options

```
LDFLAGS =  $(CFLAGS) -L. -fstack-protector -L/usr/local/lib
-L/Users/yeung/Public/Rdebug/openssl -lcrypto –lssl
$ make && make install
```

Step 3. Install debug gems

```
$ download https://rubygems.org/downloads/debase-ruby_core_source-0.10.5.gem
$ download https://rubygems.org/downloads/debase-0.2.5.beta1.gem
$ download https://rubygems.org/downloads/ruby-debug-ide-0.7.0.gem
$ export PATH=/Users/yeung/Public/Rdebug/2.x/ruby2/bin:$PATH
$ gem install --force --local debase-ruby_core_source-0.10.5.gem
$ gem install --force --local debase-0.2.5.beta1.gem
$ gem install --force --local ruby-debug-ide-0.7.0.gem
```

Step 4. Start debugging session

```
$ rdebug-ide --host 127.0.0.1 --port 1234 --dispatcher-port 1234 -- /full/path/of/main.rb
```

Add breakpoints in Speare code editor, click "Start" button on the debug toolbar of Speare code editor.
step in, step out, step next ...

Extend Speare Code Editor for mruby/c, mruby, Ruby and Ruby on Rails Debugging

To add some customised scripts to support better Ruby debugging, please download the guide document and following the description in it.