Ruby : A Dynamic Object Oriented Scripting Language

Gautham Anil
gautham_anil@cse.iitb.ac.in

November 29, 2004

Introduction to Interactive Ruby

Interactive Ruby

Interactive Ruby is similar to an operating system shell. It allows the user to execute each line of a Ruby program as we type.

Syntax of irb

irb syntax

Typically irb can be used to run a script and see its results step-by-step. The syntax for irb invocation is :

```
irb [ irb-options ] [ ruby-script ] [ script-options ]
```

Example

Example

```
$ irb
irb(main):001:0> a = 1 +
irb(main):002:0* 2 * 3 /
irb(main):003:0* 4 % 5
2
irb(main):004:0> 2+2
4
irb(main):005:0> def test
irb(main):006:1> puts "Hello, world!"
irb(main):007:1> end
nil
irb(main):008:0> test
Hello, world!
nil
irb(main):009:0> exit
```

Useful tool

Useful tool

irb is a great learning tool. It is very handy in understanding the script. Also if Ruby was built with GNU Readline support then we can go back to previous then we can change our previous statements using the scroll keys.

Command Line Options

Command Line Options

Some of the command line options are :

- -f : Suppress reading ~/.irbrc
- -m : Math Mode { Fraction and Matrix support is available}
- -r module : Same as “ruby -r”
- --readline : Use the Readline extention module
- --prompt : Switch prompt mode
- -v, --version : Version of irb

Initialization

Initialization

Initialization is done by loading one of the following following files in the given order :

- ~/.irbrc
- .irbrc
- irb.rc
- _irbrc
- $irbrc

In the initialization file we can set the commonly used options and required ruby statements.

Configuration

Configuration Values

- IRB.conf[:IRB_NAME] = "irb"
- IRB.conf[:MATH_MODE] = false
- IRB.conf[:USE_TRACER] = false
- IRB.conf[:USE_LOADER] = false
- IRB.conf[:IGNORE_SIGINT] = true
- IRB.conf[:IGNORE_EOF] = false
- IRB.conf[:INSPECT_MODE] = nil
- IRB.conf[:IRB_RC] = nil

Configuration

Configuration Values

- IRB.conf[:BACK_TRACE_LIMIT] = 16
- IRB.conf[:USE_LOADER] = false
- IRB.conf[:USE_READLINE] = nil
- IRB.conf[:USE_TRACER] = false
- IRB.conf[:IGNORE_SIGINT] = true
- IRB.conf[:IGNORE_EOF] = false
- IRB.conf[:PROMPT_MODE] = :DEFAULT
- IRB.conf[:PROMPT] = { ... }
- IRB.conf[:DEBUG_LEVEL] = 0
- IRB.conf[:VERBOSE] = true

Commands

Control Commands

There are many commands to control the irb session. Some of them are :

- exit, quit, irb_exit : These are used to quit the current irb session or subsession
- conf : Displays current configuration
- conf.debug_level = N : Sets the debug level of irb.
- conf.ignore_eof = true/false : Specifies behaviour when end of file recieved on input
- conf.ignore_sigint : true/false : Specifies behaviour of ^C
- conf.math_mode : Displays whether Ruby is in math mode or not

Prompt Configuration

Prompt Configuration

We can also change the prompt that irb uses. Sets of prompt are stored in the prompt hash :
IRB.conf [:PROMPT]

Example

This is entered in the .irbrc file or directly at the irb prompt. It establishes a new prompt mode called PERSONAL.

```
IRB.conf[:PROMPT][:PERSONAL] = { # name of prompt mode
 :PROMPT\_I => "...",       # normal prompt
 :PROMPT\_S => "...",       # prompt for continuing strings
 :PROMPT\_C => "...",       # prompt for continuing statement
 :RETURN => "  ==>%s\n"     # format to return value
}
```

Prompt Configuration

Usage

The above prompt mode is used as follows :

```
$ irb --prompt PERSONAL
```

or

set the following configuration value :

```
IRB.conf[:PROMPT_MODE] = :PERSONAL
```

RTAGS

rtags

rtags is a command used to create TAGS file for use with emacs or vi editor. By default it creates TAG file for emacs. Using -vi option we can override it.

```
rtags [-vi] [files]
```

XMP

xmp

irb's xmp is an "Example Printer". i.e a pretty-printer that shows the value of each expression as it is run.

Example xmp usage

```
require "irb/xmp"
xmp <<END
artist = "Doc Severinsen"
artist
END
```

produces:

```
[pwd:/tc/work/ruby/ProgrammingRuby/latex]
artist = "Doc Severinsen"
==> "Doc Severinsen"
    artist
==> "Doc Severinsen"
```

Frame Class

Frame Class

The 'IRB::Frame' class represents the interpreter's stackand allows easy access to the 'Binding' environment in effect at different stack levels.

IRB::Frame.top(n = 0) :
Returns a Binding for the nth context from the top. The 0th context is topmost , most recent frame.

IRB::Frame.bottom(n = 0) :
Returns a Binding for the nth context from the bottom. The 0th context is the bottommost, initial frame.

IRB::Frame.sender :
Returns the object (the sender) that invoked the current method.

Thank You