Ruby developers use intensely meta programming techniques, to create domain specific languages, to develop frameworks or to do a little bit of ruby magic. The ruby browser describe in this proposal is a tool that help developers to inspect their classes.

```
require 'rubygems'
require 'ruby2ruby'

class A
  def bar(bar)
    :bar
  end
end

eval("class A
  def foo; :foo; end
end ")

# We want to inspect the source code
# of class A
put Ruby2Ruby.translate(A)

# We want to inspect the source of the
# method foo in the class A
puts A.new.method(:foo).to_ruby
```

(see in pastie)

The goal of this project is create an interactive interface that make easy to inspect (at run-time) the state of the environment. The project don’t pretend to replace the irb console, or to be a complete IDE, the goal is improve the irb with old and new features as a gem, and create a graphical environment, but all the features could be use programatically, or used by different front end.

The ruby browser is inspired in Squeak Browser, a graphical interface that allow to navigate throw all the classes and objects inside of the Squeak virtual machine

When I’m writing a program in Smalltalk, I start a Workspace. Type in some code and evaluate it and open it up in an Inspector. As the program grows, I start refactoring the code in the Workspace into classes and inspecting it. At this point, I have an inspector open on some object. I can go in and modify or add methods in the class and evaluate them in the Inspector. I can open up inspectors on the results of those method calls, and so on.
Functionality and Mock ups

[Disclaimer] : The image of attached to this document are used as example and mock up to show the idea.

Ruby Browser will be a tool that provides methods to inspect the state of a Ruby environment. One of the goals of Mole is display the way that meta programming affect to our system in a human readable presentation.

The list of feature is:
- List all the instances of a class.
- List all the method of a live object.
- Display the source code of a class or a method.
- List all the classes that are currently load in the ObjectSpace.
- List all the senders that use a method.
- List all the implementers of a method.
- List all the method that are overridden by a method.
- Evaluate code in a irb integrated console.
- Change a method code using the browser.
- Inspecting Collections of objects.
- Hierarchy browser.
- Object browser.

Another layout approach using GTK:
classComment: aString

*Store the comment, aString, associated with the object that refers to the receiver.*

classComment:_aString

{aString (isKindOf: RemoteString)
 ifTrue: [classComment:_aString]
 ifFalse: [aString == nil or [aString size = 0]]
 ifTrue: [classComment:_nil]
 ifFalse: ]

self error: use aClass classComment.*

classComment:_RemoteString newString: aString onFileNumber: 2];

*Later add priorSource and date and initial?*
The Senders and implementors methods:
The senders method will give you a list of all methods that may use the selected method. The Senders and implementors method works in a similar way, it lists all of the classes that implement a method with the same selector.

Save Versions of the altered methods:
* When you save a new version of a method, the old one is not lost. RubyBrowser keeps all of the old versions, and to go back ("revert") to an old version. this tool is not a substitute of your version control system is utility that allow you to navigate in the change timeline that you have been doing at method level along the session.
The Method Overriding browser
The method_overriding method displays all the methods overridden by the current method.

The hierarchy browser
Display the hierarchy of the current class. Who are the parents (modules hierarchy is also support) and the childrens of a class.
Inspecting Collections of managed objects

The collection inspector described earlier in this article leverages the introspection features provided by Core Data. You can open it by double-clicking on a collection in the object browser, or programmatically.

And of course, it will be able to use Enumerable syntax to perform queries or update the state of the collection.
The object browser:
Display the current state of an Object.

=== Benefits to the Community ===

Sometimes, in the Static vs. Dynamic debate we are missing one point: the interactivity. If we can inspect faster and easier our code, we can interact, modify, and experiment with the live objects and their definitions at runtime we gain a huge productivity boost. With a collection of methods to make inspection easier and a GUI irb console, we reduce the learn curve. And for those who already don't know object oriented programming. They can have a graphical representation of the object that they created in the console.

=== Related gems : ===

* Ruby2ruby
* ParseTree
* sepr_processor