The Clone's Family Tree [Updated]

Posted on **April 16, 2016**

In Ruby, most data is accessed by reference. That is, variables contain a sort of hidden pointer to the data, not the data itself. This is very efficient but has a hidden trap. Assignment does not copy the data. It only copies the *reference*! Consider the following snippet of code:

```
a = 'foo'
b = a
a << 'bar'
puts b
```

What do you think the "puts b" statement will print? Turns out it's "foobar". Since only references were copied, when the original variable (a) was modified in the third line, both variables (a and b) were "mutated".

So how does one copy values and not just references? The Ruby programming language has two methods for duplicating data. These are "dup" and "clone". While these methods are quite useful, they both suffer from two shortcomings:

1. In Ruby, if an attempt is made to clone (or dup) an immutable data item like a number, an error occurs. The justification for this uncharacteristic strictness is not at all clear, but it does mean that the clone (or dup) operation must be applied with great care.
2. The copying process used by both clone and dup is said to be a shallow (or incomplete) copy. While the target data structure is copied, any internal data structures are not. References to those data remain aliased in the copy.

I started off to create a gem to resolve these issues. I ended up creating a family of four gems that are tailored to the exacting data copying requirements of the application. Here is a summary of those gems:

Family Chart

Depth / Action	Need to copy all data and metadata attributes?	Need to copy data only?
Only need a shallow copy?	Use the safe_clone gem. <Source>	Use the safe_dup gem. <Source>
Need a full, recursive copy?	Use the full_clone gem. <Source>	Use the full_dup gem. <Source>

Notes:

- Since none of these gems override the default clone and dup methods, the default behaviors remain available. Further, if multiple, differing requirements exists, more than one family member gem may be employed in the same project without fear of conflict.
- If multiple family gems are employed, they will each need to be installed and required into the application. See each gem's github source for details.
- Meta-data attributes include the frozen status and singleton methods. However the tainted status is always copied.

I hope you find these little gems as useful as I have found them to be. I you like them, give the code repository a star as a sign of approval!

Yours Truly

Peter Camilleri (aka Squidly Jones)

This entry was posted in **Dev Tools**, **Ruby** by **Peter Camilleri**. Bookmark the **permalink [http://teuthida-technologies.com/?p=1698]** .