code wrapped up in a module that a class can include or extend (more on those terms later). In fact, a single class can have many mixins.

**What about Inheritance?**

One area I like to use mixins is for logging. In my Rails applications I like each model to have access to a logger without using a global constant of some sort. Initially I thought about using some sort of base model class (which could inherit from ActiveRecord::Base) that would provide access to a logger. However, I didn’t really like the idea of all of my models having to extend a special class just to get a logger. Yes, they would already extend ActiveRecord::Base, so what’s the difference, right?

Well, what if I wanted to add access to a logger in other parts of my application which didn’t inherit from ActiveRecord::Base (or a special subclasss)? Ruby only supports single inheritance, so mixins seemed to be the best solution.

**Include vs Extend**

Modules can either be included or extended depending on whether the methods will be added as instance methods or class methods, respectively.