STATE MACHINES

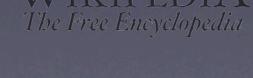
By CodeOfficer

codeofficer.com
github.com/codeofficer
twitter.com/codeofficer

State Machines?

A model of behavior composed of a finite number of states, transitions between those states, and actions. A finite state machine is an abstract model of a machine with a primitive internal memory.

http://en.wikipedia.org/wiki/Finite_state_machine



So, What's in a State Machine?

- Machine
- States
- Events
- Transitions



Machines?

- Consist of states, events and transitions that define how a state changes after an event is fired
- Model behavior for a class via an attribute of that class
- Multiple state machines can be used in a class, each one tracked by a unique attribute

States?

- Have an initial state
- Represent the value for a particular machine in an attribute of your class
- Attribute can be a value of any type. The default type in Ruby is often "String"
- Can be used to define a behavioral context for a given machine



Events?

- Define an action that transitions a machine from one state to another
- Guards can be put in place to make state transitions conditional when an action is fired

Common Uses?

- Spree for processing orders
 in_progress, new, canceled, returned, resumed, paid, shipped
- Restful Authentication
 passive, pending, active, suspended, deleted
- ActiveModel in Rails validations, callbacks, observers
- Tracks (GTD) for TODO statuses active, project_hidden, completed, deferred

Gems and Plugins

State Machine

A plugin By Aaron Pfeifer (pluginaweek) http://github.com/pluginaweek/state_machine/ http://api.pluginaweek.org/state_machine/

Acts As State Machine

A gem By Scott Barron http://github.com/rubyist/aasm/

Which state machine project will have the most github watchers by RailsConf 2009? https://opensource.inklingmarkets.com/markets/18368

A Simple Machine

(using the State Machine plugin)

```
class Light < ActiveRecord::Base
  attr_accessor :intensity
  state_machine :state, :initial =>:off do

  state :off { def intensity; 0; end }
  state :low { def intensity; 5; end }
  state :high { def intensity; 10; end }

  event :switch do
    transition :off => :low, :low => :high, :high => :off
  end
  end
end
```

```
@light = Light.new
                                  @light.state
                                                    #=> "low"
@light.state
                  #=> "off"
                                  @light.intensity #=> 5
@light.intensity
                 #=> 0
                                  @light.off?
                                              #=> false
@light.off?
             #=> true
                                  @light.can_switch? #=> true
@light.can_switch? #=> true
                                  @light.switch!
@light.switch!
              #=> true
                                  # repeat and rinse ...
```

Another Machine

(also using the State Machine plugin)

```
class Vehicle
  state_machine :initial => :parked do
    event :park do
     transition [:idling, :first_gear] => :parked
    end
    event :ignite do
     transition :stalled => same, :parked => :idling
    end
    event :idle do
     transition :first_gear => :idling
    end
    event :shift_up do
     transition :idling => :first_gear, :first_gear => :second_gear, :second_gear => :third_gear
    end
    event :shift_down do
     transition :third_gear => :second_gear, :second_gear => :first_gear
    end
    event :crash do
     transition [:first_gear, :second_gear, :third_gear] => :stalled
    end
    event :repair do
     transition :stalled => :parked
```

Machine Integrations

(for the State Machine plugin)

- Database Transactions
 every transition is wrapped within a transaction
- Automatically Saves Records
 @thing.event vs @thing.event! (bang throws an exception on fail)
- Named Scopes
 Thing.with_state(:off).all or Thing.with_state([:off, :on]).all
- Validation Rrrors
 @thing.errors.full_messages # => ["State cannot ... via :xxx from :yyy"]
- Observers observers can hook into before/after callbacks for events and generic transitions

Textmate Bundles

Auto completion for machines, events, transitions and callbacks

- State Machine
 http://github.com/drnic/ruby-state-machine-tmbundle/
- Acts As State Machine
 http://github.com/levicole/acts-as-state-machine-tm-bundle/