Análise comparativa de ferramentas de Workflow para Ruby

Engenharia de Processos e Requisitos

Eduardo Mauricio Barbiero

Sistemas de Informação - Católica de Santa Catarina

Joinville, Dezembro de 2015

Resumo

Análise comparativa entre dois FrameWorks desenvolvidos com Ruby, para auxiliar no Fluxo de trabalho de um determinado negócio. Será feita uma breve descrição sobre os dois frameworks "Workflow" e "State Machines", que possuem algumas particularidades e demonstrado em forma de tabela a representação de funcionalidades e diferenças.

1 Workflow

Workflow é uma API inspirada na "máquina de estados finitos" para a modelagem e interagir com o que temos a tendência de se referir como "fluxo de trabalho".

Uma grande remeça de informações sobre o negócio, tende a envolver conceitos de fluxo de trabalho, bem como, essa biblioteca tende a expressar que esses conceitos sejam o mais claro possível. É utilizada uma terminologia semelhante encontrada na "teoria de máquina de estado".

Um fluxo de trabalho possui sempre um estado. Ele só pode estar em um estado de cada vez, porém, quando ele muda de estado, chamamos isso de transição. As transições de um evento, ocorrem de modo que cada transição ocorra, onde caso haja alguma falha na execução dessa transição, seja executado uma outra remessa de código, que possa ou não corrigir esse problema. Desse modo, esses transições podem gerar vários eventos, e qualquer evento pode gerar uma outra transição ou uma nova ação.

Aplicando esses conceitos a um caso, digamos que uma cidade possui alguns problemas e devem ser resolvido com alguns passos, até que chegue a um determinado setor de uma empresa. Dessa forma, um problema é diagnosticado, passado a uma analista de problemas até chegar ao responsável pelo setor. Após isso, deve ser enviada a uma determinada área para execução ou reparação do problema, vejamos na utilização da API:

```
class Problemas
 include Workflow
 workflow do
   state :new do
     event :submit, :transitions_to => :aguardando_revisao
   end

   state :aguardando_revisao do
     event :review, :transitions_to => :sendo_revisado
   end

   state :sendo_revisado do
     event :accept, :transitions_to => :accepted
     event :reject, :transitions_to => :rejected
   end

   state :accepted
   state :rejected
 end
end
```

Nesse caso, o uso ficaria dessa forma:

```
problema = Problemas.new
problema.accepted? # => false
problema.new? # => true
```

Ou seja, é verificado se o problema foi aceito, e verifica se é um evento novo.

Uma outra forma de verificar esse status, é chamando o objeto "`current_state`" que retorna um objeto de estados do workflow atual.

```
problema.current_state
=> #<Workflow::State:0x7f1e3d6731f0 @events={
  :submit=>#<Workflow::Event:0x7f1e3d6730d8 @action=nil,
    @transitions_to=:aguardando_revisao, @name=:submit, @meta={}>},
  name:new, meta{}
```

Agora precisamos saber, se o problema foi aceito, e onde se encontra nossa review:

```
problema.current_state < :accepted
=> true
problema.current_state >= :accepted
=> false
problema.current_state.between? :aguardando_revisao, :rejected
=> true
```

Nesse caso, podemos observar, que ele se encontra "`aguardando_revisao`", e desse modo ele ainda é "rejeitado".

Agora nós podemos chamar o evento "submit", que faz a transição para o estado "`aguardando_revisao`":

```
problema.submit !
problema.aguardando_revisao? # = > True
```

Esses eventos, são na verdade métodos de instancia de um fluxo e trabalho, que dependendo do estado onde você está, ele vai conter um conjunto de eventos diferentes para serem usandos e para auxiliar na transição para outros estados.

É fácil verificar se uma certa transição foi feita, a partir do evento atual, basta verificar pelo método "`problema.can_submit?`", se essa transição foi submetida do estado atual.

2 State Machines

State Machines é uma API baseada na antiga "State Machine" que foi descontinuada a algum tempo, porém leva em consideração a grande parte de sua tecnologia.

A State Machines torná seu workflow operante e simples de gerenciar o comportamento de uma classe, simplifica o projeto, introduzindo as várias partes de uma máquina de estado real, incluindo estados, eventos, transições e retornos de chamada. No entanto, a API é projetado para ser tão simples que você não precisa mesmo de saber o que é uma máquina de estado.

Vejamos um exemplo simples de aplicação, utilizando como base um Problema de infraestrutura na cidade, onde só poderia ser possível transitar pela rua, se todos os problemas fossem resolvidos.

Exemplificando:

```
class Infraestrutura
  state_machine :state, initial: :transitar do
    after_transition on: :reparar, :transitar
    after_failure on: :transitando, :necessita_reparacao

    state :necessita_reparacao do
          def reparacao
            @reparacao = true
          end
          def avisar_prefeitura
            #executa evento para avisar a prefeitura
          end
        end

    state :transitar do
      transition stalled: same, :transitando
    end

        state :transitando do
          def transitando
            @transitando = true
      end
```

```
        end

        event :reparar do
          def parar
            @transitando = false
          end
          def reparado
            @reparacao = false
          end
        end

        def initialize
          @transitando = false
          @reparacao = false
        end

        def parado?
      @transitando
    end
  end
end
```

Desse modo, podemos observar, que o estado "`necessita_reparacao`", é o fim do fluxo de trabalho, onde ele irá lançar uma notificação para a prefeitura, para resolução do problema. Vejamos como usar na prática.

```
infra = Infraestrutura.new
infra.state # transitando

infra.transitando? # true

# [#<StateMachines:Transition attribute=:state
# from="transitar" from_name=":transitar" to="transitando"
# to_name=:transitando>]
infra.state_transitions

# suponhamos que o metodo transitando, falhe
```

```
infra.state # necessita_reparacao
infra.reparar
infra.state_event # [:reparar]

# vejamos que quando o evento de reparacao e chamado
# o mesmo ja faz a transicao para o estando "transitando"
# ao finalizar
infra.state # transitando
```

---

3 Comparação

A seguir será feita uma comparação entre as duas tecnologias.

	Ruby	Maleável	Rastreável	Interface	Tutoriais	Escalabilidade	Processos
WorkFlow	1.9.x	+1	+1	-1	+1	-1	+1
State Machines	2.x	+1	+1	-1	-1	+1	+1

4 Conclusão

Pelo fato do gem Workflow ser extremamente simples de ser usado e aplicado, torna-se uma boa escolha, quando se trata de um sistema de não tanta regra de negócio sobre o fluxo de trabalho, pela complexidade do código quando se trata de persistência. Ao contrário da State Machines que traz uma boa camada para desenvolvimento sobre persistência.

Com base nisso vale destacar que o programador deve se adequar no que realmente se precisa e a que melhor se encaixa na regra de negócio. A robustez da State Machines com relação a persistência, ou a simplicidade da Workflow para desenvolvimento de uma api/app de simples manutenção.

5 Referências

https://github.com/geekq/workflow. Acesso em 30/11/2015.
https://github.com/state-machines/state_machines. Acesso em 01/12/2015.
https://github.com/pluginaweek/state_machine. Acesso em 01/12/2015.