

ruote

# A Ruby workflow engine

Wes Gamble aka “Bison Consulting”  
10 August 2010

# What'll we discuss

- What is ruote?
- So what?
- ruote basics
- Advanced topics

# What is ruote?



# What is workflow?

- “A workflow consists of a sequence of connected steps”
- “A workflow is a model to represent real work for further assessment”
- “The term workflow is used in computer programming to capture and develop human-to-machine interaction”
- The things that happen to stuff

# ruote is...

- a workflow engine written in Ruby (e.g. a persistent manager that knows “who” is participating in a workflow and how they pass work to each other)
- an “operating system for business processes”
- not a state machine library
- derived from OpenWFE (Java), then ported to OpenWFeru
- maintained by John Mettraux (@jmettraux)

# So what?

- We have AASM - what more do we need?
- AASM and ruote are two different, yet complementary approaches
- AASM attaches behavior to objects (object is primary e.g. “noun-y”)
- ruote manages the processes that involve objects (process is primary e.g. “verb-y”)
- As state transition logic becomes more complex, a proper workflow engine becomes more helpful

# Why do you need ruote?

## You have...

- well - defined processes that have complex logical transitions
- a need for concurrency
- a need to persist state of workflow for a long time
- diverse participants in the workflow



# ruote basics: Components

- Process Definition: how is process executed (“series of steps”)
  - made up of Process Expressions (“steps”)

<http://ruote.rubyforge.org/expressions.html>

- Participants: Actors in the workflow (people, programs, ?)
- Workitems
- Engine: Provides process instances (RT)

# ruote basics:

## Process Definition

- “documents that describe how a [business] process should orchestrate the flow of work among participants”
- Specified in Ruby DSL, XML, tree
- AST is created when def. is parsed at launch time (process instance)
- Different “versions” of a process definition can be in play depending on editing and process instance launches
- Sub-processes can be specified and referred to in other processes

# ruote basics:

## Sequential process

```
Ruote.process_definition :name => 'my_def' do
  sequence do
    participant 'alice'
    participant 'bravo'
  end
end
```

- One time through
- Two participants: 'alice' and 'bravo'

AST:

```
["define", {"name"=>"my_def"}, [{"sequence",
  {}, [{"participant", {"ref"=>"alice"}, []},
  ["participant", {"ref"=>"bravo"}, []]]]]
```

# ruote basics:

## Concurrent process w/looping

```
pdef = Ruote.process_definition :name => 'work' do
  cursor do
    concurrence do
      reviewer1
      reviewer2
    end
    editor
    rewind :if => '${f:not_ok}' # back to the reviewers if editor not happy
    publish # the document
  end
end
```

- Participants: 'reviewerX', 'editor', 'publish'
- Loop via “cursor” and rewind
- Parallel tasks via “concurrence”
- Conditional processing based on workitem contents (:if => '\${f:not\_ok}')

# ruote basics:

# Participants

- These are the actors in a workflow
- In general, participants “ask” for work, and then “reply” after they have modified the workitem. (But some don’t explicitly do this, like “block” participants).
- They are of specialized types which gives them special behavior
- Participants get their own thread to do processing
- Examples:
  - Block: Provides a block that processes a received workitem and then immediately returns it
  - SMTP: Encapsulates mail server metadata to ease sending of email
  - Storage: To allow for participants to store their workitems using the same storage mechanism as the engine
- Others

# ruote basics:

# Workitems

- These are the pieces of data that are actually passed around between the participants
- Glorified hash
- Field values within the workitems can be used to
- When you launch a process instance, you pass it a workitem

# Example block participant

```
alpha = engine.register_participant :alpha do |workitem|  
  workitem.fields['time'] = Time.now  
end
```

- The workitem is automatically passed to the block from the process instance
- Stuff happens
- The workitem is automatically returned to the process instance when the block ends

# ruote basics: Engine

- The engine is the Ruby runtime that keeps track of all of the workflows
- Workers talk to engine to handle processing tasks
- Multiple storage options:
  - HashStorage: in-memory
  - FsStorage: files
  - DmStorage: DataMapper
  - CouchStorage: CouchDB
  - RedisStorage: Redis
  - BsStorage: Beanstalk



# ruote basics:

## Running a workflow

*SETUP - do this once, usually*

*Initialize the engine:*

```
storage = Ruote::FsStorage.new('./tmp/ruote_work')  
worker = Ruote::Worker.new(storage)  
@engine = Ruote::Engine.new(worker)
```

*Create a process definition:*

```
@pdef = Ruote.process_definition :name => 'xyz'...
```

*Register participants (what kind of participant):*

```
@participants = @engine.register_participant '+',  
Ruote::StorageParticipant
```

*RUNTIME - do this on demand as needed*

*Start a process instance:*

```
wfid = @engine.launch(@pdef, {wi_key: wi_value})
```

# Demo



# Advanced Topics:

## Fancy Participants

- AMQP: places items in a queue (also a listener)
- jig (rufus-jig HTTP client): POSTs received workitems as JSON to a server and stores answer back
- Rufus::Decision::Participant: Use a CSV based decision table (local or Web-addressable) to drive workitem updates (allow business logic to be updated independent of code)

# Example

## Rufus::Decision table

```
Rufus::Decision::Participant, :table => %{  
  in:topic,in:region,out:team_member  
  sports,europe,Alice  
  sports,,Bob  
  finance,america,Charly  
  finance,europe,Donald  
  finance,,Ernest  
  politics,asia,Fujio  
  politics,america,Gilbert  
  politics,,Henry  
  ,,Zach  
})
```

# Advanced Topics: Other Stuff

- ruote - kit:

A RESTful wrapper around the ruote workflow engine, built as a modular sinatra application

- ruote - fluo:

Graphically represent a ruote process definition

# Resources

- <http://ruote.rubyforge.org/>
- <http://groups.google.com/group/openwferu-users>
- <http://www.opensource.co.za/2009/08/25/ruote-in-20-minutes-video/> (ruote + AASM)
- <http://ruote.rubyforge.org/rdoc/>
- <http://jmettraux.wordpress.com/>
- <http://jmettraux.wordpress.com/2009/07/03/state-machine-workflow-engine/>