Introduction to Ruby Topic Maps

Benjamin Bock

http://rtm.rubyforge.org
Whassup today?

Ruby Topic Maps

RTM+Ruby on Rails
I’m your Edutainer

Benjamin Bock

Ruby Coder

Topic Maps Researcher

Author of Ruby Topic Maps
Schedule

⇒ Ruby Topic Maps

☐ RTM on Rails
Ruby Topic Maps (RTM)

Topic Maps Engine

Easy Programming Interface

(TMAPI++)

Database Backend
RTM is for you

Optimized for programmer happiness
( like Rails and Shoes are )

Direct access to properties
( productivity++ )* 

Beautiful code: easy to read and understand
( Ruby has low line-noise )

* Btw, the ++ operator known from other languages is not available in Ruby, use += 1 instead.
RTM is opinionated

Convention over Configuration
→ just start using it

Do what I want - functionality
→ don’t query objects, just use them

Standards compliant and a little more
→ predefined associations
RTM is structured

**Back-end** based on ActiveRecord*

**Programming** interface is a wrapper layer

**Integrated** Ruby-style query language

* That’s the model part (i.e. relational database mapper) from Ruby on Rails
RTM is written in Ruby

Ruby is object-oriented but also procedural and functional

everything is an object

dynamic and (but) strong typed
Install ok?

# ruby --version
ruby 1.8.6 (2007-09-24 patchlevel 111)
  [i386-mswin32]
ruby 1.8.6 (2008-08-11 patchlevel 287)
  [x86_64-linux]
# rails --version
Rails 2.1.1
Install ok? (2)

```ruby
# gem search rtm
rtm (0.1.4)
# irb
irb(main):001:0> require 'rtm'
=> true
irb(main):002:0> RTM.connect
=> true
```
begin
    puts "Do you need a Ruby Introduction?"
    x = gets
end until x =~ /(yes|no)/i

open("Ruby Intro") if x=~/yes/i
Loading

# loading the Ruby Topic Maps library
require 'rtm'

# Connecting to a back-end
RTM.connect  # Memory
RTM.connect_sqlite3("mydb.sqlite3")
RTM.connect_mysql("database_name", "user_name", "password", "host")
Initialization

# generate database schema
RTM.generate_database

# enable SQL statement logging
RTM.log

# create a TopicMap
tm = RTM.create "http://tmra.de/tm1/"
Getting Topics (overview)

# get a topic using its identifiers:
# item identifier:
t1 = tm.get("item-identifier")
# subject identifier:
t2 = tm.get("http://psi.example.org/t2")
# subject locator:
t3 = tm.get("=http://rtm.rubyforge.org")
Getting Topics I

# get by item identifier
`t1 = tm.get("item-identifier")`

# * use relative IRIs
# * returns nil if not found
# for using absolute IRI:
`t1 = tm.by_item_identifier("absolute:/:item-identifier")`

# * the latter might be TopicMapsConstruct, too
Getting Topics II

# get by subject identifier

```ruby
t1 = tm.get("absolute:/identifier")
```

# * use absolute IRIs
# * returns nil if not found

# or use the direct method:
```
t1 = tm.topic_by_subject_identifier("absolute:/subject-identifier")
```
Getting Topics III

# get by subject locator

t1 = tm.get("=http://
    rtm.rubyforge.org")

# * similar to subject identifier
# * prefix with ":=
# or use the direct method:

t1 =
    tm.topic_by_subject_locator( "http://
        //rtm.rubyforge.org")

# * no prefix needed here
Creating Topics (overview)

# similar to getting, add ! to method
# item identifier:
t1 = tm.get!("item-identifier")
# subject identifier:
t2 = tm.get!("http://psi.example.org/t2")
# subject locator:
t3 = tm.get!("=http://rtm.rubyforge.org")
Creating Topics (continued)

```
# similar to getting, add ! to method
# item identifier:
t1 = tm.topic_by_item_identifier!
    ( "item_identifier"
# => always returns a Topic
# subject identifier:
t2 = tm.topic_by_subject_identifier!
    ( "http://psi.example.org/t2"
# subject locator:
t3 = tm.topic_subject_locator( "http://rtm.rubyforge.org"
# no =
```
Setting and getting occurrences

using ["occurrence-type-get-id"]

t1["age"] = 25
# * creates a new occurrence
# * sets type to topic_map.get!("age")
# * sets value to 25 and datatype to int
t1["age"]
# * fetches all occurrences of t1 with given type
... and (almost) the same for names

Using 

```
using ["-name-type-get-id"], like occurrence but prefixed with "-"

# using ["occurrence-type-get-id"]
t1["-firstname"] = "Benjamin"

# no type -> default TMDM name type
t1["-" ] = "Benjamin Bock"
```
Creating TopicMapConstructs

# create a new Topic
```
t = tm.create_topic    # see also: get!(ref)
```

# create a new Association
```
a = tm.create_association
```

# create AssociationRoles
```
r = a.cr "player", RTM::PSI[:type]
```
Association’s perspective

TMAPI

Association Type

Role Type

Role Player

Role

roles

type

Role Type

Role Player

Role

roles

topic_map

Topic Map
Association’s perspective

TMAPI++

Association Type

Role Type

Role Player

role_types

roles

role_players

parent

Topic Map
Role’s perspective

TMAPI

Association Type

Role Type

type

Role Player

player

association

Role Type

type

Role Player

player

association

player
Role’s perspective

TMAPI++

Role Type

Association Type

Role Type

Role Player

Role Player

player

player

counterparts

counterplayers

type

type
Topic’s perspective

TMAPI++

Association Type

Role Type

roles_typed

roles_typed

Role Player

roles*

roles*

counterparts

counterpart players

* roles can also be called roles_played
Navigation

# getting a topic
t = tm.get!("my-topic")

# get a list of all its variant’s values
defname.variants.map { |v| v.value }

# there is a shortcut for simple mapping:
defname.variants.value
Characteristics-access

# getting it’s default name
defname = t["-"].first
# there might be many, so it is a set,
# we take the first TopicName object we get
fname = t["-firstname"].first
age_occurrence = t["age"].first
Creating Characteristics

# setting the default name
# this creates a new one:
t['-'] = "Benjamin Bock"

# this changes the existing one:
t['-'].first.value = "Benjamin Bock"

t['age'] = 25
Querying

# Get all Topics without name
m.t.select { |t| t.n.size == 0 }

# Get all Association types
ti = m.a.map { |a| a.type }.uniq
Import and Export

# Import an XTM 2.0 file
RTM.from_xtm2(io_stream, "base_locator")

# Or use much faster libxml
RTM.from_xtm2lx(file_name, "base_locator")

# Export a complete topic map
xml_string = m.to_xtm2
Find out more

No docs yet 😞
Browse:
ruby/lib/ruby/gems/1.8/gems/rtm-x.y.z/

We’ll do a walkthrough...
Now: go on playing!

Questions?

http://rtm.rubyforge.org
Schedule

☑ Ruby Topic Maps

⇒ RTM on Rails
Ruby on Rails

- web application framework
- model
  - ActiveRecord
  - RTM
  - ActiveTM
- view
  - embedded Ruby
  - HAML, RJS, …
- controller
Install ok?

# ruby --version
ruby 1.8.6 (2007-09-24 patchlevel 111) [i386-mswin32]
ruby 1.8.6 (2008-08-11 patchlevel 287) [x86_64-linux]

# rails --version
Rails 2.1.1
Install ok? (2)

# gem search rtm
rtm (0.1.4)
# irb

irb(main):001:0> require 'rtm'
=> true

irb(main):002:0> RTM.connect
=> true
Creating an app

On your command line do:

```
rails projectname
```

This generates a skeleton application
Directory structure

- app
  - controllers
  - helpers
  - models
  - views
- config
- db
- lib
- public
vendor/plugins

- There are lots of helpful plugins, e.g. on github.com
- Copy them into this directory and they are available in your app.
- Each plugin can
  - Have a “normal“ gem-folder-structure
  - Have a app-folder-structure, like a standalone rails app
Environments

- development
  - Reloads stuff every time
  - Keeps database
  - Slow, but useful
- Production
  - Caches a lot
- Test
The server

Launch the web server:
   ruby script/server

Defaults:
   • Runs locally on Port 3000 (switch: -p)
   • Uses development environment (-e)
The console

Launch the (debugging) console:
  ruby script/console

• This is IRB + your Rails app
• Environment analog to server
• This is *not* the same as the running server, it just shares code, settings, db
“Implant“ RTM

In file config/environment.rb add:

```ruby
require 'rtm'
```

at the end. Same for other libraries...

In the console (which needs a restart!)
```bash
run: RTM.generate_database
```
Rake – Make for Ruby

• Rails provides a set of rake tasks
• Find them with
  rake --tasks

• Needed e.g. for migrations, but not (yet) for RTM / ActiveTM
Generators

- They generate parameterized code
- Show help + list available generators:
  ruby script/generate
- Show help for a specific generator:
  ruby script/generate generatorname
- Add more parameters to actually run it
- Find more:
  gem search –r generator
Migrations (Generator)

- Updates the database schema
- A DSL covering SQL Create/Alter statements
- `script/generate migration` 
  
  name var:integer var2:string var3:text  
  var4:date var5:datetime ...
- `rake db:migrate / rake db:rollback`
- Not needed with RTM, but helpful
MovieDB

Let’s have a look...
Thank you!

Questions?