UNIVERSITÄT LEIPZIG

Getting started with

Ruby Topic Maps

http://rtm.rubyforge.org

Benjamin Bock

Third International Conference on Topic Maps Research and Applications 2007-10-12

Schedule

- ⇨ Ruby, Rails and RTM
- ☐ Real Source Code
- ☐ Scalability and Performance

Introduction

Web 2.0 is about integration

Ruby and Ruby on Rails are Big Players there

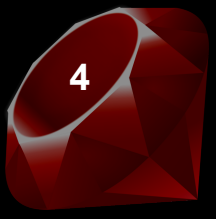
Topic Maps nonexistent in the Ruby world

Why Ruby?

interpreted, object-oriented programming

features procedural and functional paradigm

dynamically and/but strongly typed



Ideas behind RoR

Optimized for programmer happiness

Writing beautiful source code

Convention over Configuration

Goals of RTM

Usable out of the box

Direct access

Type less, reach more

Current Status

Quick & easy installation

Ready for use memory and database back-end

XTM 2.0 import and export

Internal Structure

Back-end based on Active Record

Main implementation is a wrapper layer

Mixed-in modules for serialization, merging, extended API

API Gimmicks

Direct use of String references to Topics

Enumerable Sets provide query language

Zero overhead command shortcuts

Schedule

- ☑ Ruby, Rails and RTM
- ⇨ Real Source Code
- ☐ Scalability and Performance

Loading

```ruby
# loading the Ruby Topic Maps library
require 'rtm'

# Connecting to a back-end
RTM.connect # Memory

RTM.connect_mysql("database_name",
  "user_name", "password", "host")
```

Initialisation

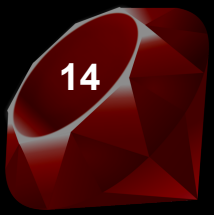
```
# generate database schema
RTM.generate_database

# enable SQL statement logging
RTM.log

# create a TopicMap
tm = RTM.create "http://tmra.de/tm1/"
```

Creation

```
# create a new Topic
t = tm.create_topic

# create a new Association
a = tm.create_association

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

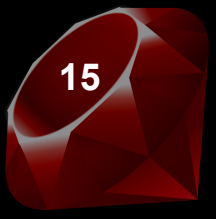
Navigation

```
# get a (random) TopicName
n = tm.get!("player").names.first

# get all scoped Variants of the
  first scoping Topic found
vs = n.scope.first.scoped_variants

# get array of Variant values
vs.map {|v| v.value}
```



Querying

```
# Get all Topics without name
nn = m.t.select {|t| t.n.size == 0 }
# Get all Association types
ti = m.a.map {|a| a.type }.uniq
# oblige Robert Barta
m.t.each {|t| t.v.each {|v|
  if v.datatype == PSI[:string]
    t.cn v.p.to_hash.merge(v.to_hash)
  else
    t.co v.p.to_hash.merge(v.to_hash).merge(
     :type => PSI[:variant_name])
  end
  v.remove
}}
```



Import and Export

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

# Export a complete topic map
xml_string = m.to_xtm2

# Export other formats
m.to_jtm; m.to_yaml; …
```

Schedule

- ☑ Ruby, Rails and RTM
- ☑ Real Source Code
- ⇨ Scalability and Performance

Yes, but... does it scale?

Speed?

NO! unfortunately not (yet)

Scaling manpower!

Big optimization potential.

Performance

SQLite3:

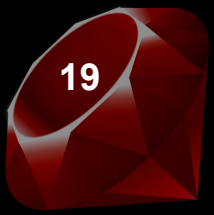
100 Topics in 13s (committing each)

Memory:

Creates 10,000 empty Topics in <30s

10,000 empty Associations in <20s

300KB XTM 2.0 takes 45s to import



Schedule

- ☑ Ruby, Rails and RTM
- ☑ Real Source Code
- ☑ Scalability and Performance

Outlook

Needs to prove itself in real life

Higher Level API in sight

Community wanted!

`http://rtm.rubyforge.org`

Thank you!

Questions?