

ActiveRDF: object-oriented RDF in Ruby

Eyal Oren Renaud Delbru

DERI

June 9, 2006

Overview

- ▶ most programming languages object-oriented
- ▶ most RDF APIs triple-oriented
- ▶ → programming RDF unnecessarily complex

- ▶ some semantic differences between RDF and OO
- ▶ in scripting languages we can solve these differences
- ▶ ActiveRDF: object-oriented RDF in Ruby

Challenges in object-oriented RDF

- ▶ semi-structured data: RDF resources do not need to belong to single class with fixed behaviour.
- ▶ inheritance: RDFS allows multiple inheritance
- ▶ type semantics: with open world semantics, class membership and object behaviour can change during runtime
- ▶ runtime flexibility: RDF data and schema can change continuously (data integration), should not depend on fixed schema.

Other approaches (wrongly) assume existence of a schema, stability of schema, and strict conformance to schema.

Addressing challenges with scripting language

Scripting language: dynamic typing, meta-programming, interpreted execution, runtime reflection.

- ▶ semi-structured data: dynamic type system allows objects with different behaviour than their classes
- ▶ inheritance: meta-programming allows override of single-inheritance
- ▶ type semantics: meta-programming and multi-inheritance allow changing class membership at runtime
- ▶ flexibility: interpreted execution and reflection can provide virtual API at runtime

ActiveRDF features

- ▶ complete RDF manipulation through Ruby objects, using dataset terminology: `renaud.age`
- ▶ dynamic finders based on available data: `find_by_firstname`
- ▶ schema independence: classes, objects, and methods are created on-the-fly from apparent structure in data
- ▶ connect to various kinds of data stores (YARS, Redland, SPARQL, *Sesame*, *Jena*)
- ▶ Ruby on Rails integration

Example ActiveRDF

```
# connect to YARS database containing FOAF information
NodeFactory.connection
  :adapter => :yars, :host => :browserdf.org

# create or load person, display friends
eyal = Person.create 'http://activerdf.org/eyal'
print eyal.knows

# find person by name, name and age, and by keyword in name
renaud = Person.find_by_firstname 'renaud'
jack = Person.find_by_firstname_and_age 'jack', '30'
all_johns = Person.find_by_keyword_firstname 'john'
```

Manipulating RDF data

```
# create or load person
renaud = Person.create 'http://activerdf.org/renaud'

# runtime reflection for supported methods
renaud.methods → [firstname, lastname, knows]

# get and set properties
renaud.firstname → 'renaud'
renaud.lastname = 'Delbru'

# save into database (generates RDF update query)
renaud.save

# print each friend (standard Ruby closure)
renaud.knows.each do |friend|
  print friend.firstName
end
```

Manipulate schema

- ▶ if schema available, classes and methods are generated automatically
- ▶ schema can be extended manually

```
class Researcher < Person
  # we set the URI of this class
  set_class_uri 'http://activerdf.org/researcher/'

  # we add three predicates to this class
  add_predicate 'http://activerdf.org/topic'

  # we specify the attribute name
  add_predicate 'http://purl.org/dc/terms/title', 'dc_title'
end

# create a researcher, display his methods
renaud = Researcher.create 'http://activerdf.org/renaud'
renaud.methods
→ [firstname, lastname, topic, dc_title, ...]
```

Summary

- ▶ ActiveRDF: object-oriented RDF in Ruby
- ▶ released 0.1, soon 0.2, Windows and Linux
- ▶ collaboration DERI, PUC-Rio, volunteer contributions
- ▶ used in several projects: <http://browserdf.org>
- ▶ open source, GPL

- ▶ more information: <http://activerdf.org>