SEMITAR

Ruby << RDF | OWL

Programming with the Semantic Web

Rich Kilmer, InfoEther LLC

Overview

- Quick introduction to Ruby
- Semitar library
- Processing RDF & OWL with Semitar
- From object-oriented to property-based programming
- Future directions of Semitar

Ruby

- Created by Yukihiro Matsumoto (1993)
- Dynamic object-oriented scripting language
  - Everything is an object (a la smalltalk)
  - No scalers
  - Classes/Objects are open
- Powerful text processing
  - Regular expressions with Perl 5 engine
- Lambdas & continuations [a la lisp]
- Easily extensible in C
- Extensive libraries

Semitar - RDF

- Model-centric RDF library
- Sources
  - URL Source
  - File Source
- Parsers
  - Pure Ruby N-triples/rdf-xml Parsers
  - Native extension wrapper for libraptor
- Generators
  - N-triples
- Query Engine
  - RDQL Inspired

Sample Semitar RDF Usage

```
require ‘semitar’
model = Semitar.new_rdf_model

model.load_file “file:tself.owl”, “rdfxml”

model.add_standard_namespaces
model.add_namespaces(
  'off' => 'http://www.daml.org/2001/10/office/office#',
  'troy' => 'http://www.daml.org/people/tself/tself#'
)

matches = model.query(:desk, :office) do
  where [:desk,   "<rdf:type>",      "<off:Desk>"],
        [:desk,   "<off:location>", :office],
        [:office, "<rdf:type>",      "<off:Office>"]
  filter { desk.uri.include?('desk1') }
end

matches.each do |match|
  puts “Desk = #{match.desk}, office = #{match.office}”
end
```

Semitar - OWL

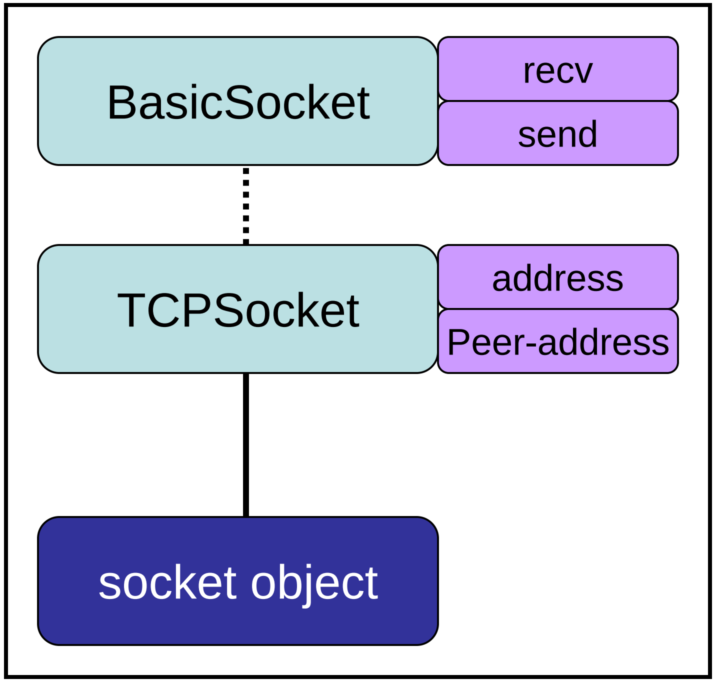
- Dynamic extension to RDF model
- OWL Classes
  - Named, anonymous, restrictions, axioms, complete class axioms, advanced constructors
- OWL Properties
  - Object Properties
  - DataType Properties
  - Annotation Properties
  - Property axioms
- OWL Individuals
  - Axioms, properties, types
- Validation (coming soon)
  - Ontology/Individuals

Sample Semitar OWL Usage

```
require ‘semitar’
model = Semitar.new_rdf_model

model.load_file “file:ebiquity.owl”

model.include_owl
model.parse_owl_ontologies

model.each_owl_class do |klass|
  puts klass
end

model.each_object_property do |op|
  puts property
  puts “  Ranges:”
  op.ranges.each {|range| puts “    #{range}”}
  puts “  Domains:”
  op.domains.each {|domain| puts “    #{domain}”}
end
```

From Object-Oriented Programming

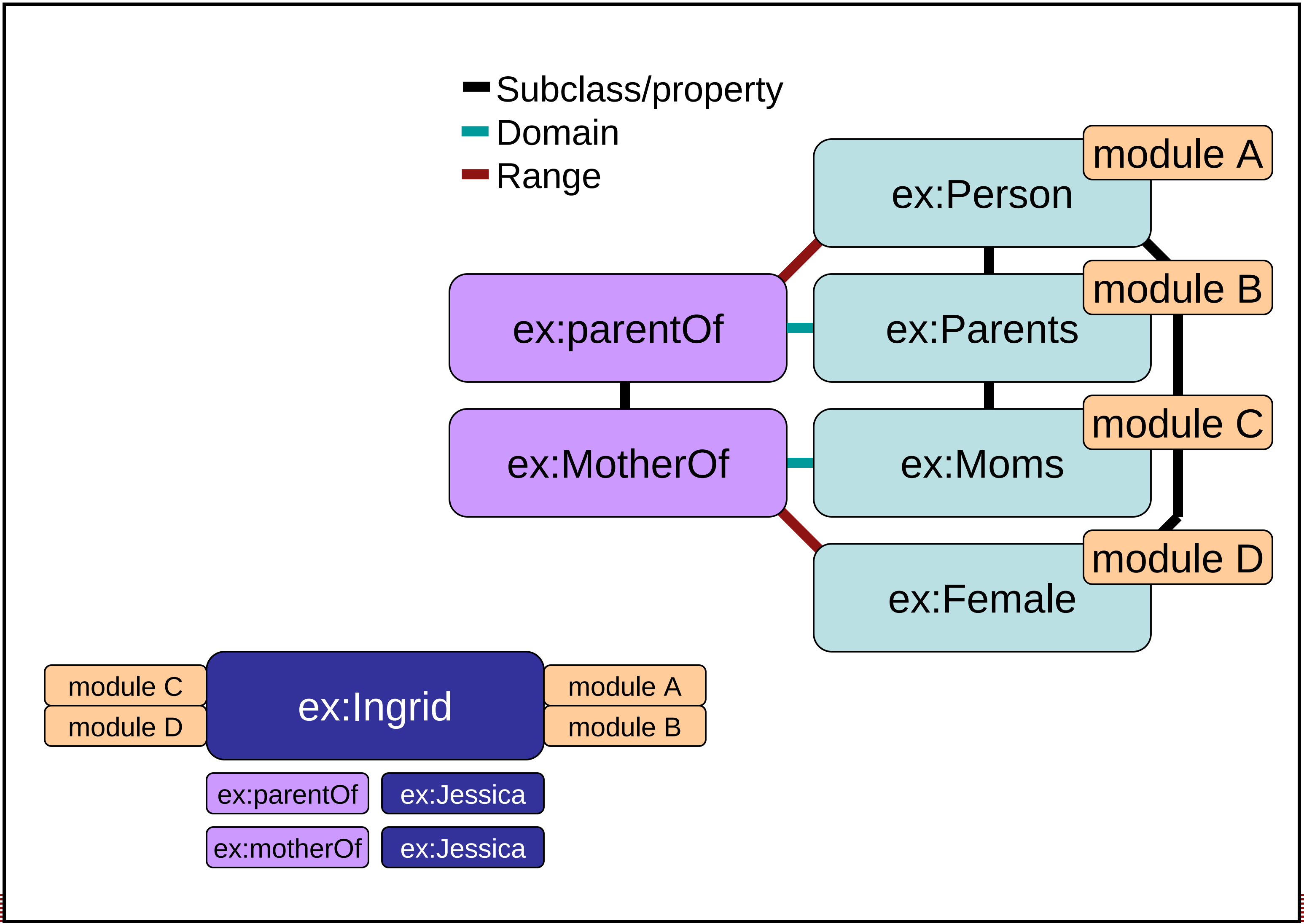
- ‘Classic’ object-oriented programming
  - Class based
    - Java, C++, Ruby
  - Prototype based
    - Self, Javascript, IO
- Operation-centric ontology design
  - Methods exist in the context of a Class
  - Encapsulation rules the day



To Property-Based Programming

- OWL ontologies
    - Class membership is dynamic
        - Asserted through <rdf:type>
        - Inferred based on properties and/or axioms
        - An object’s classes change based on ‘knowledge’
    - Properties are fully-namespaced and separate ‘objects’
- Structure-centric ontology design
- ‘Behavior’ is not expressed
- Toward property-based programming model
    - Dynamic class capabilities of OWL
    - Mixing in of behaviors (methods) based on changing memberships at runtime

Property-based Programming



Future Directions of Semitar

- Add unit testing suite
  - Use the RDF/OWL test documents
- RDF
  - Generation of RDF-XML
  - RDF Schema (normalize properties model)
- OWL/RDFS query engine
- Expand Property-based programming ideas
  - Runtime engine
    - Persistence
    - Distribution
  - Application examples

SEMITAR

Ruby << RDF | OWL

Questions?

Rich Kilmer, InfoEther LLC

rich@infoether.com