Jena: A Semantic Web Framework for Java

Reporter

C.F.Liao (廖峻鋒)

May 17,2007

Intelligent Space

國立台灣大學資訊工程研究所 智慧型空間實驗室

About Me

- Education
  - Ph.D. Candidate, CSIE ,NTU
  - Advisor: Prof. Li-Chen Fu
  - Research interests: Middleware for the smart living spaces.
- Professional Services
  - SCJP / SCWCD
  - Lecturer, SL-750, Learning Services, Sun Microsystems Taiwan, Inc.
  - Columnist (Java EE), RUN! PC Magazine
  - Reviewer, Core JSF CHT edition
  - ...

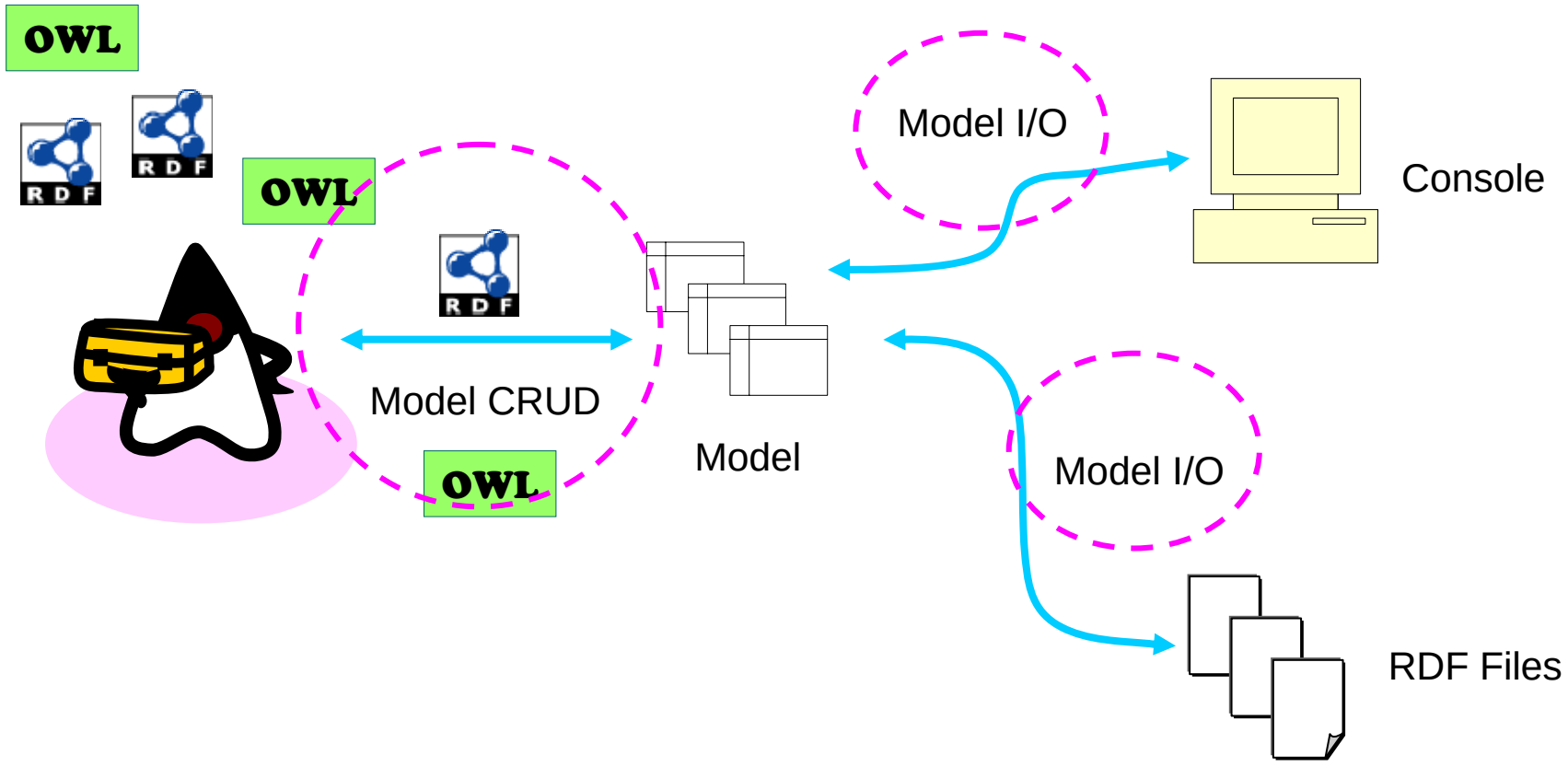
Outline

- Introduction
- Installing and Running Jena
- RDF Model Operations
- Inference Mechanisms
- (Optional)
  - SPARQL
  - Joseki
- Conclusion

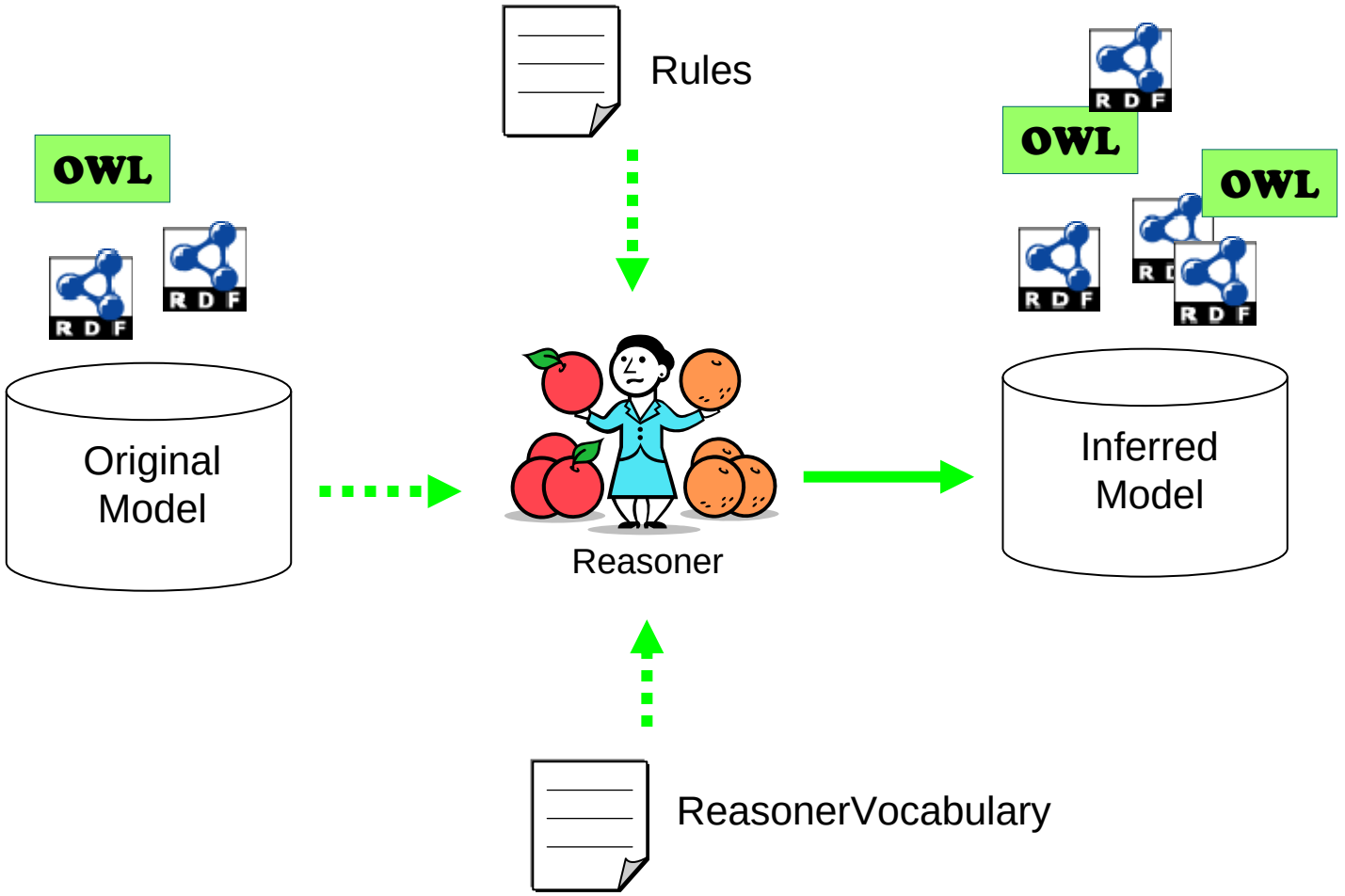
Introduction

- What is Jena?
  - An open source semantic web framework written in Java
- Jena is composed of
  - RDF Processing API
  - OWL Processing API
  - A rule-based reasoning engine
  - SPARQL query engine

RDF and OWL Processing API

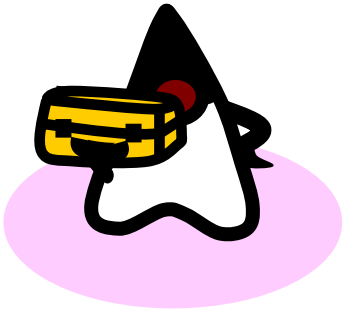


Jena Inference Mechanism

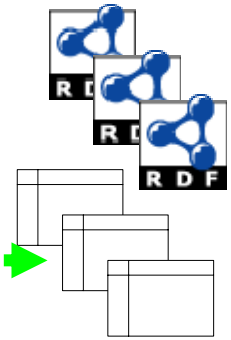


SPARQL (W3C Standard)

SELECT ?x WHERE { ?x locatedIn classroom}



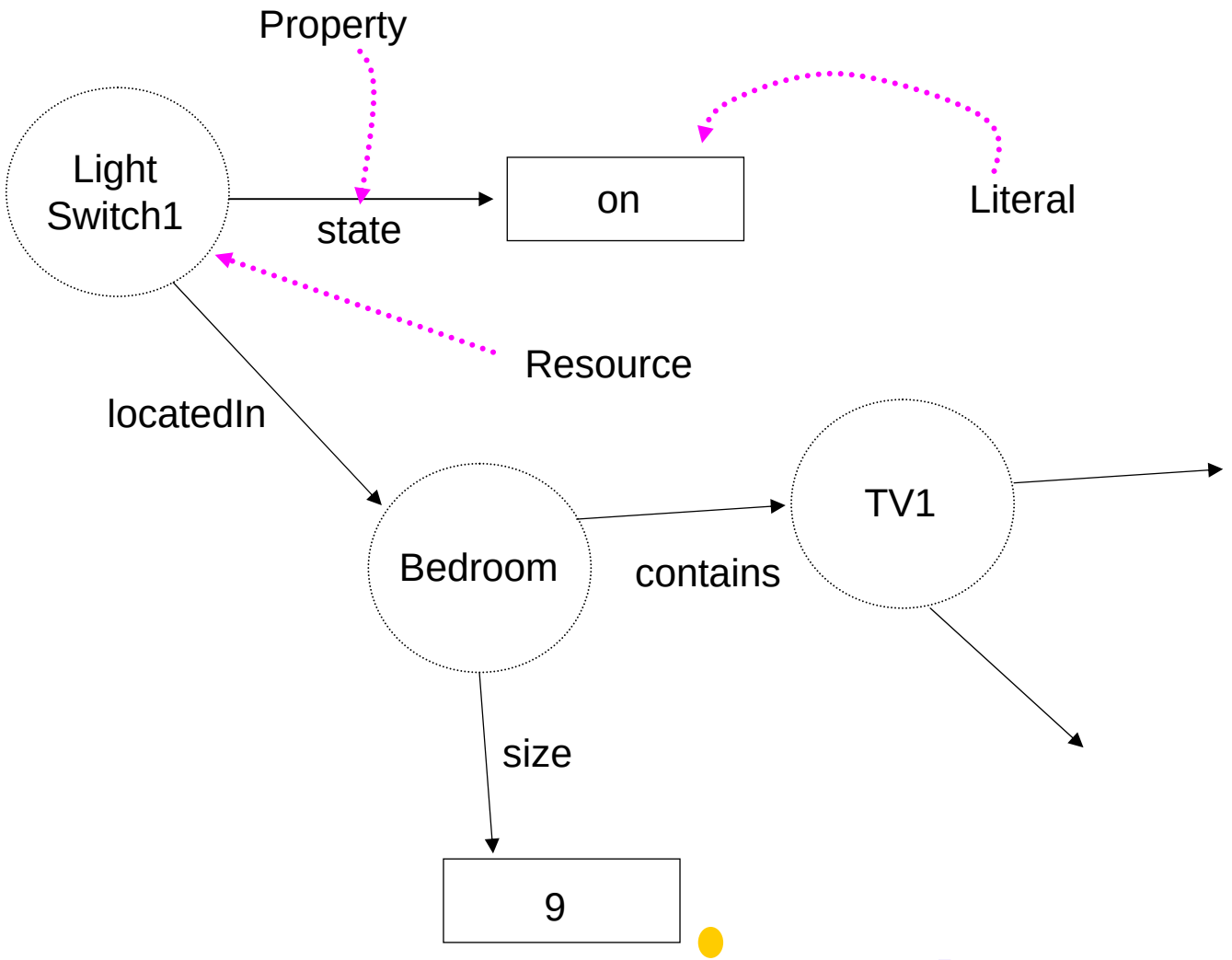
SPARQL Query Engine



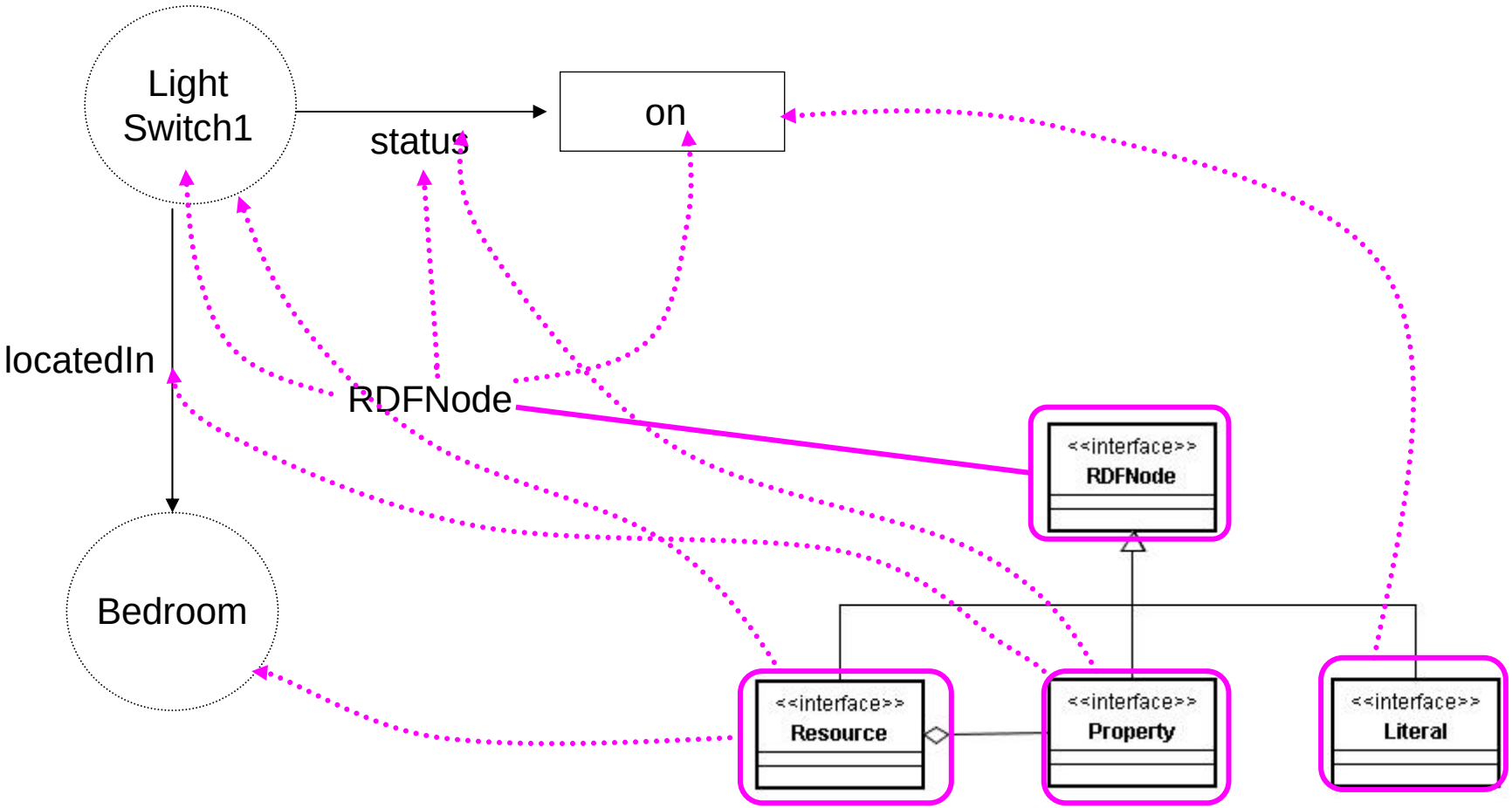
Model

?x = {Jane, Mary, and Peter}

Review: The RDF Network



Key Abstractions



Browse the Jena Javadoc

Recommended Development Environment

- JDK 1.5 +
- Eclipse 3.2 +, JST (J2EE Standard Tools)
- MySQL 4.x (optional)

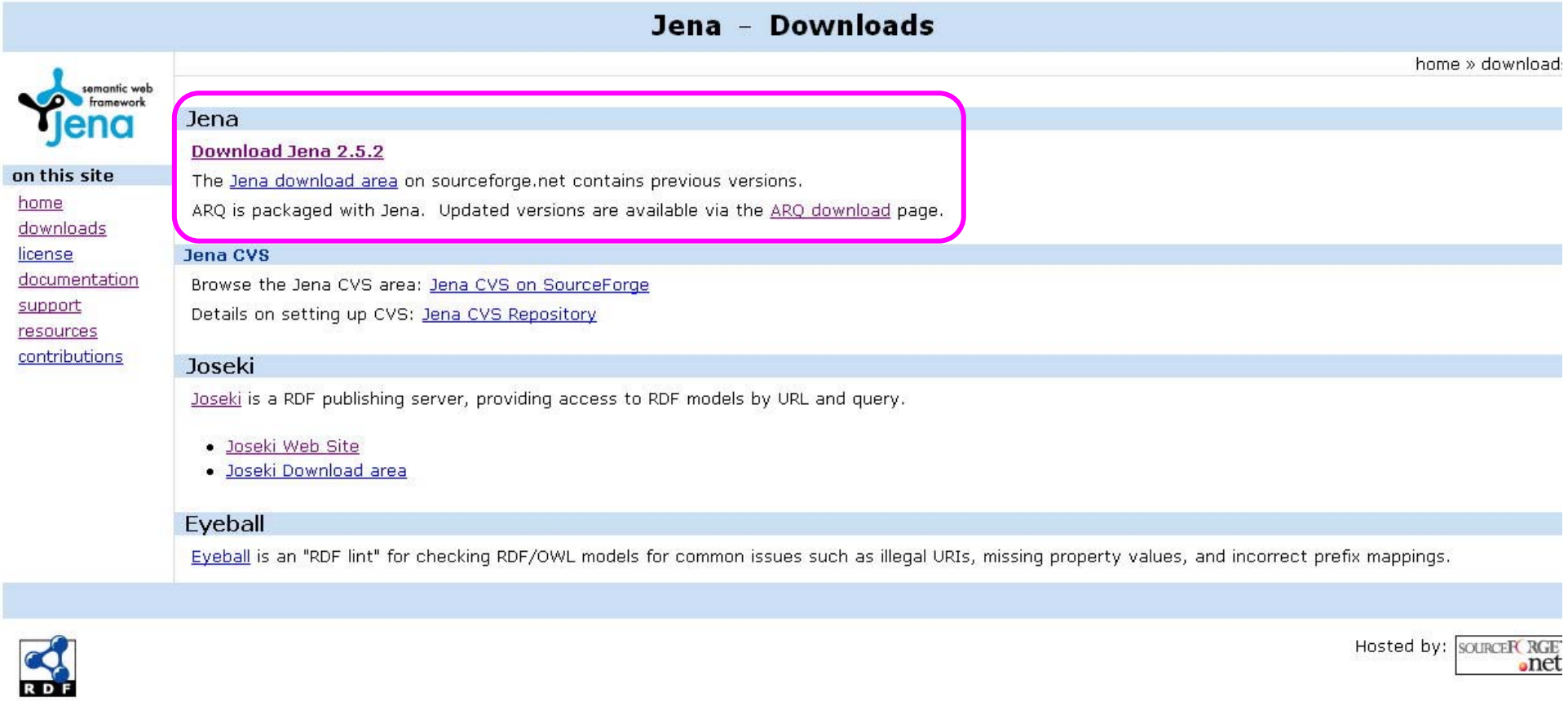
Installing and Running Jena

demo

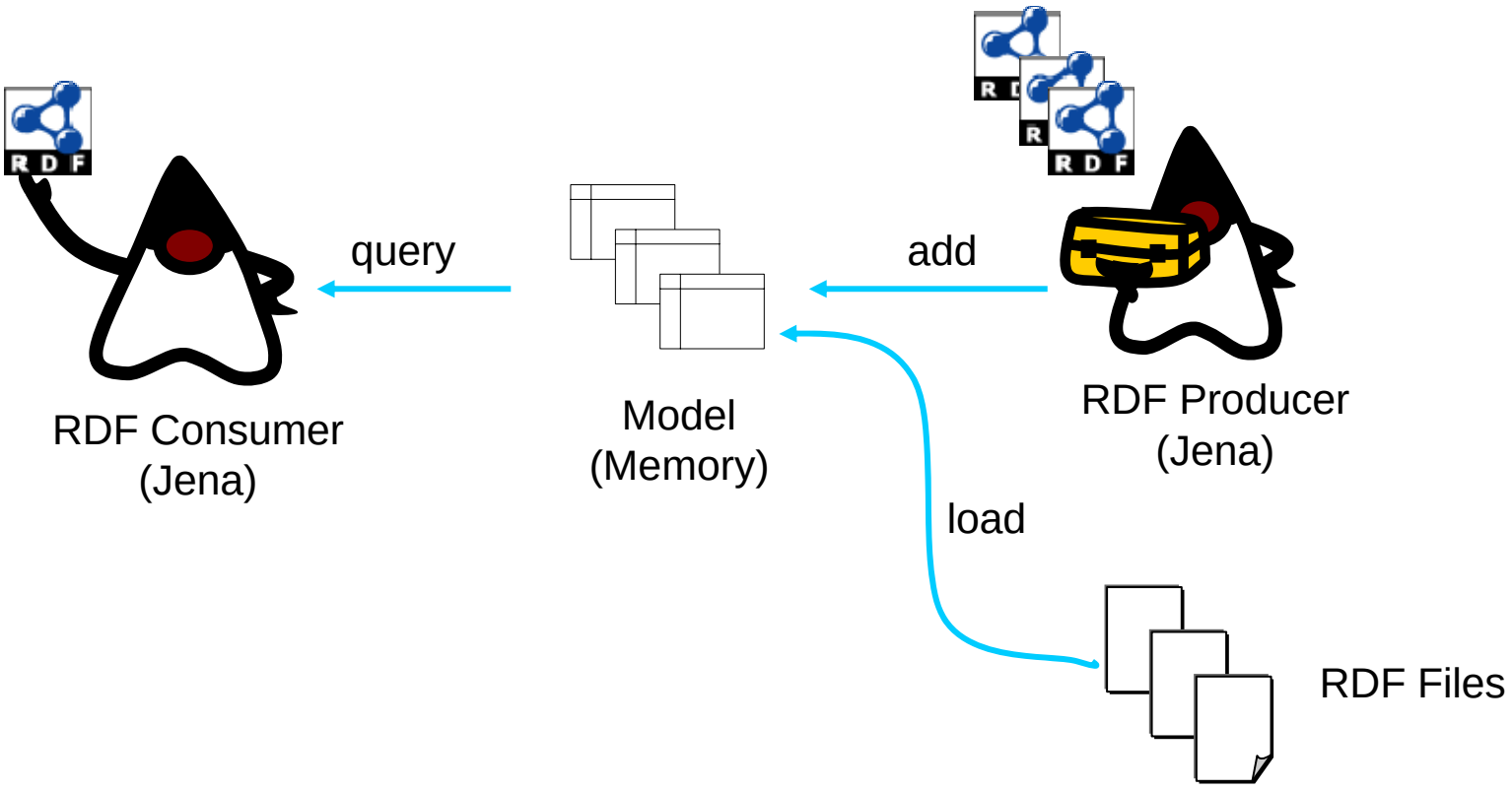
1. Download JDK 5
2. Download and install Eclipse
3. Download Jena
4. Tuning your Eclipse
5. Create a java project
6. Append Jena libraries to your classpath
7. Use Jena API to create some RDFs

Downloading Jena

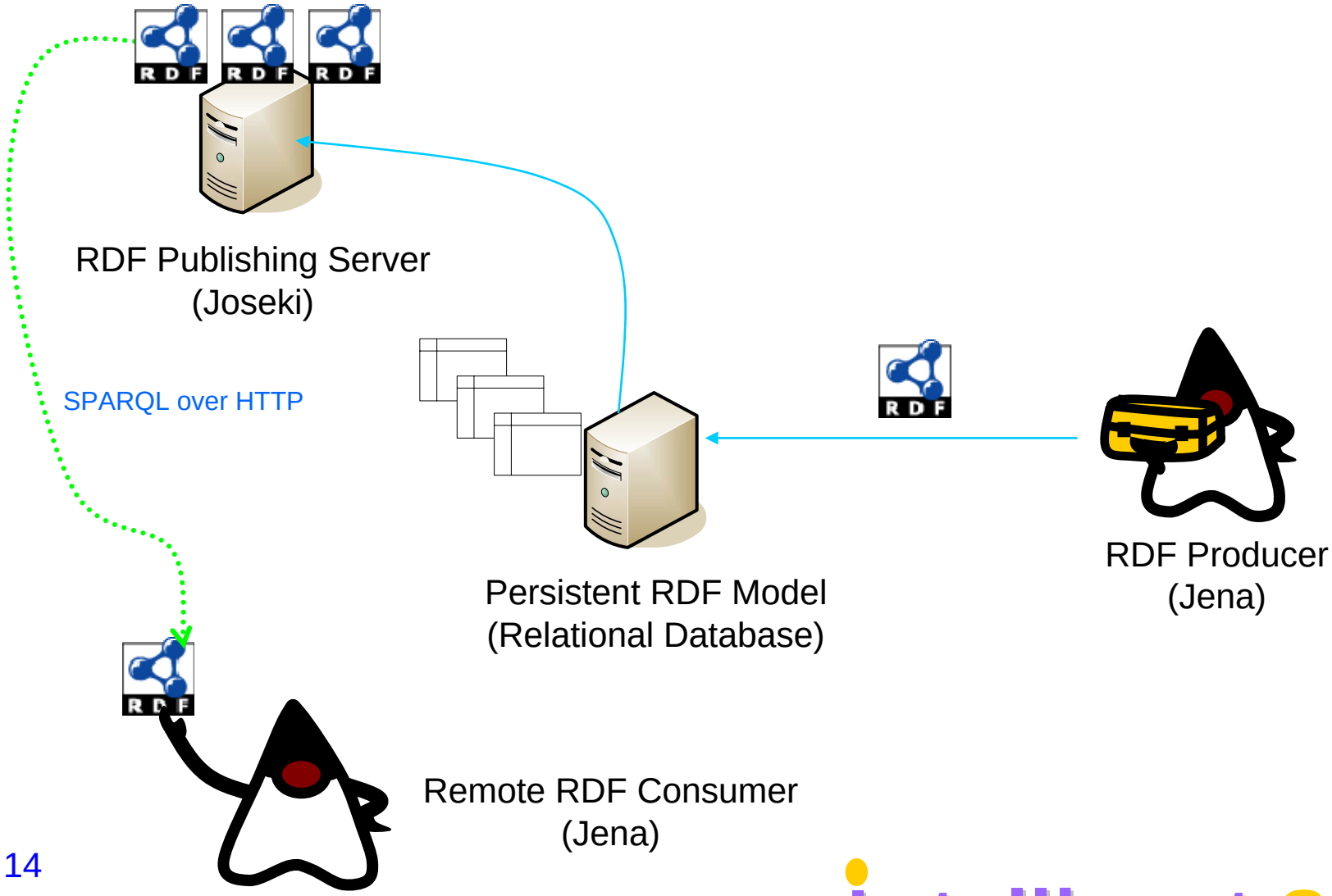
- http://jena.sourceforge.net/



Typical Jena Usage Scenario



Advanced Jena Usage Scenario

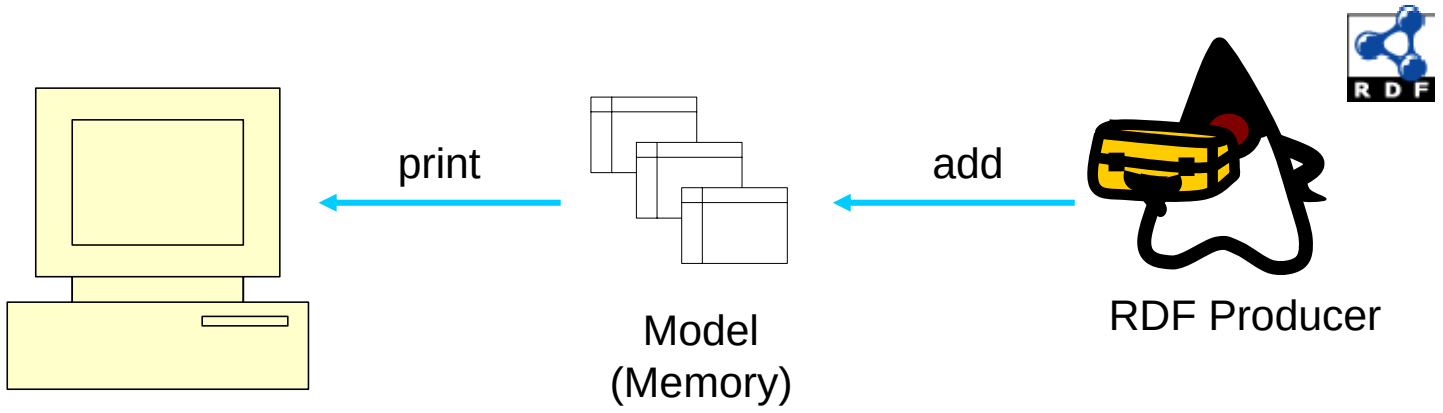


Lab

Creating a Simple RDF Model

Objectives

- Creating a simple RDF statement
- Placing this RDF into a Model
- Dumping the Model to system console



Creating RDF

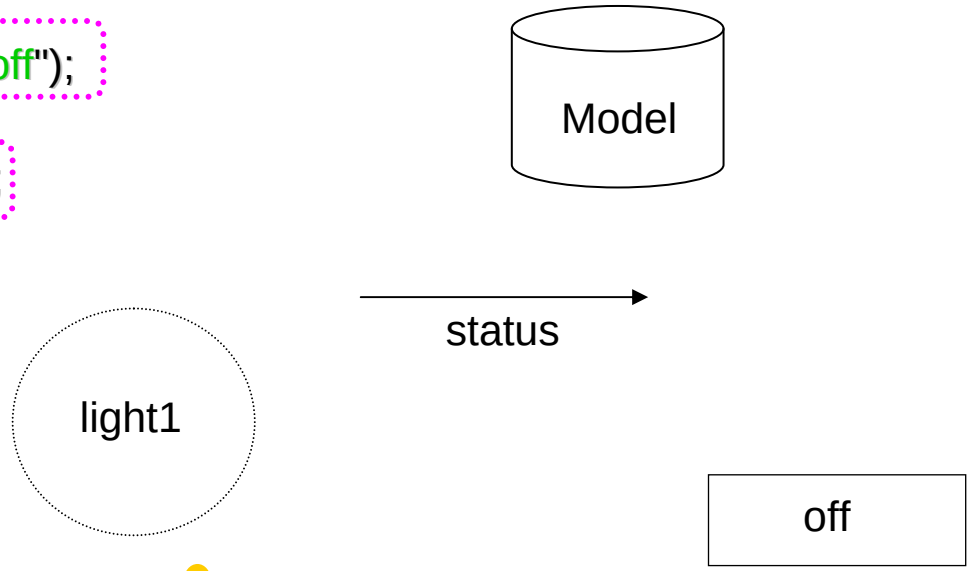
- 2 ways
  - Using Jena API
  - Load from file

Creating a RDF statement

ex1 demo

```
// declare URI prefix
String ns    = "http://www.ispace.tw/smarthome#";

Model model = ModelFactory.createDefaultModel();

Resource light = model.createResource(ns+"light1");

Property lightStatus = model.createProperty(ns+"status");

Literal statusValue = model.createLiteral("off");

light.addProperty(lightStatus, statusValue);

model.write(System.out,"N-TRIPLE");
```

(light,status,off)

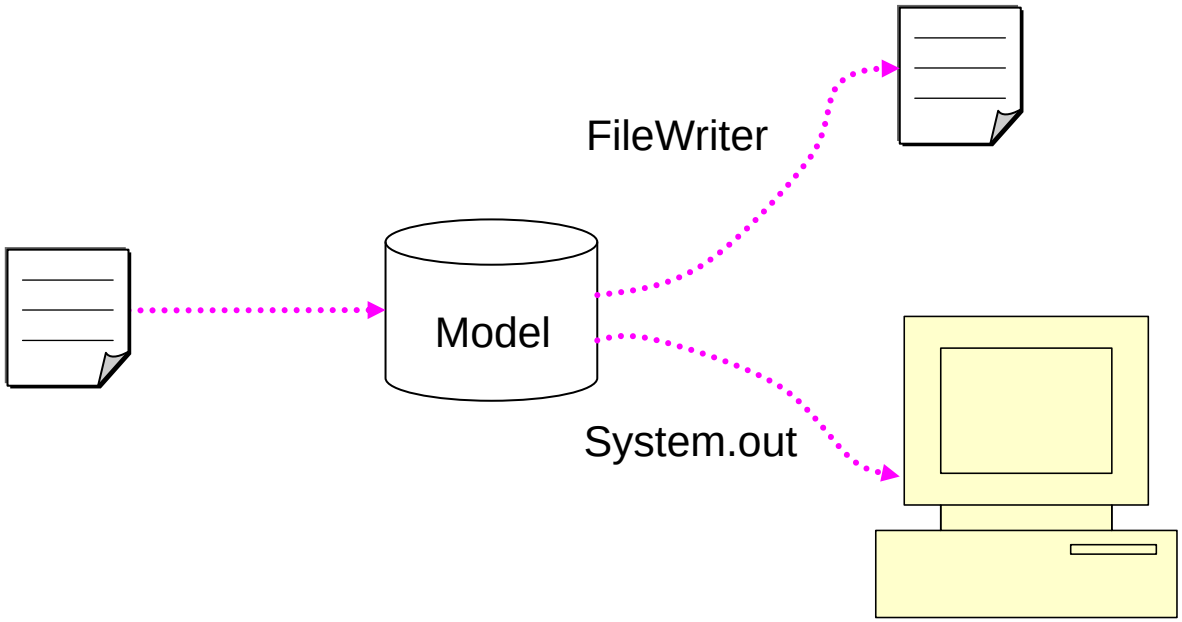


Lab

Model I/O

Model I/O

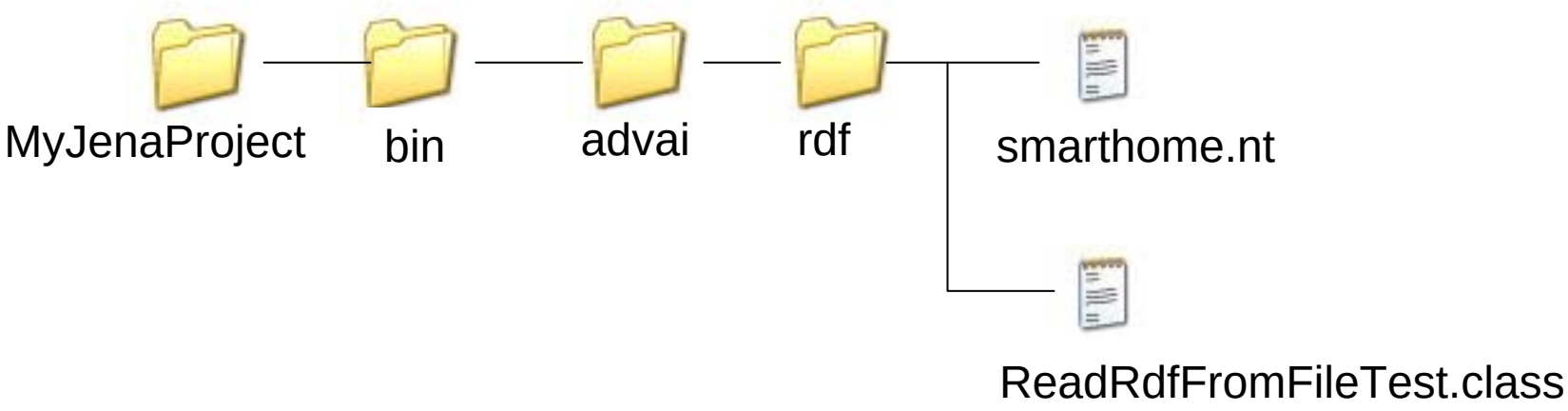
```
model.read("file:./bin/advai/rdf/smarthome.nt", "N-TRIPLE");
model.write(new FileWriter("test.nt"),"N-TRIPLE");
```



Loading RDF from an External File

ex2 demo

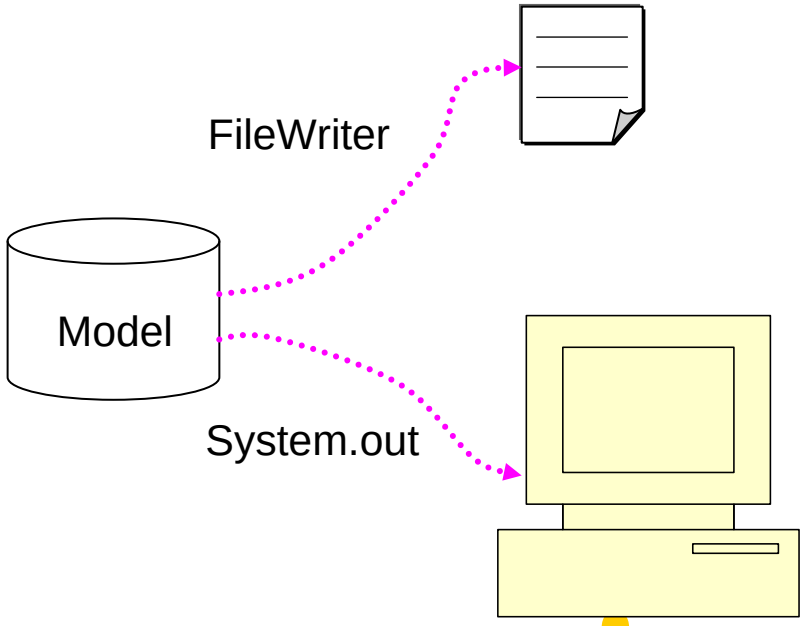
```
Model model =
        FileManager.get()
                   .loadModel(
                    “file:./bin/advai/rdf/smarthome.nt",
                    "N-TRIPLE");
```



Dumping the Model

```
model.write(System.out,"N-TRIPLE");
model.write(System.out,"N3");
model.write(new FileWriter("test.nt"),“RDF/XML");
model.write(new FileWriter("test.nt"),“RDF/XML-ABBREV");
```

ex3 demo

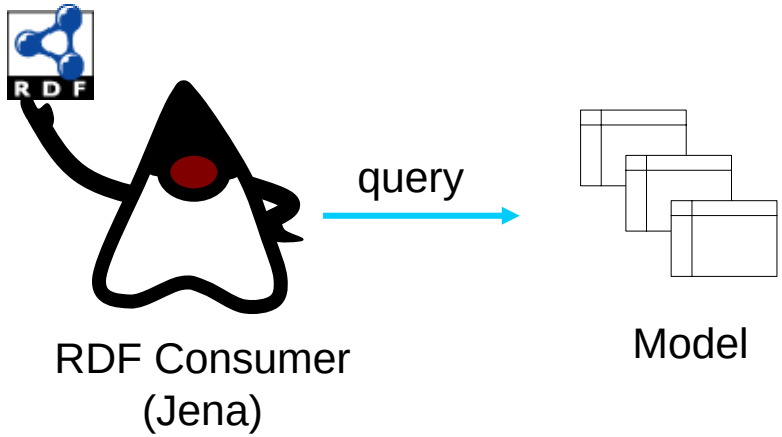


Lab

Model CRUD

Model CRUD

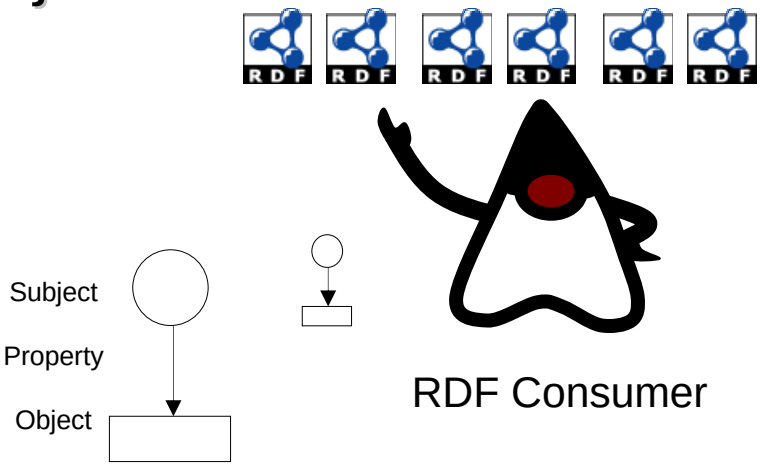
- Create
- Retrieve
- Update
- Delete

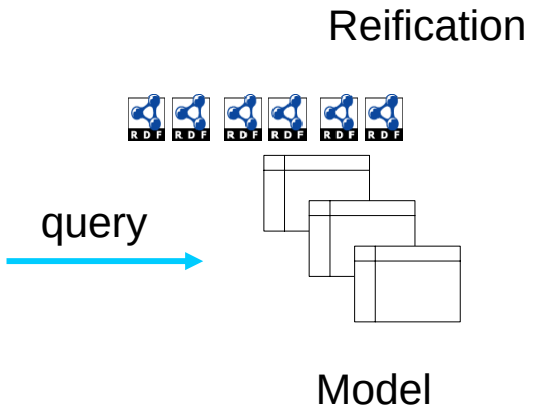


Navigating a Model

ex4 demo

```
for(Iterator it = model.listStatements();it.hasNext();)
{
    Statement stmt = (Statement) it.next();
    System.out.println(stmt.getSubject().getLocalName());
}
```



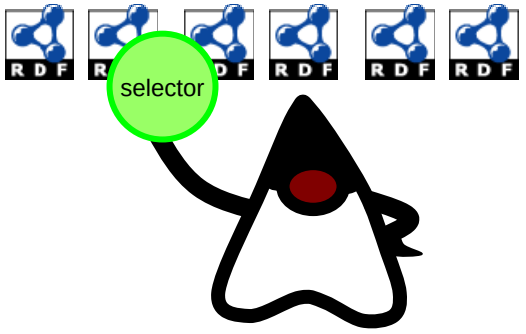


Querying Model

ex5 demo

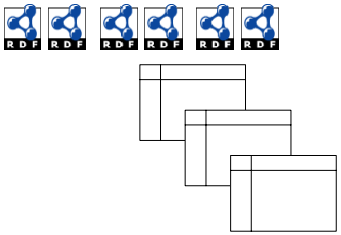
“ Select * from MODEL where SUBJECT=‘Jane’ ”

```
Resource jane =  model.getResource("http://...Jane");

Selector selector =
        new SimpleSelector(jane, (Property)null, (Resource)null);

for(Iterator it = model.listStatements(selector);it.hasNext();){   …}
```



RDF Consumer

Reification



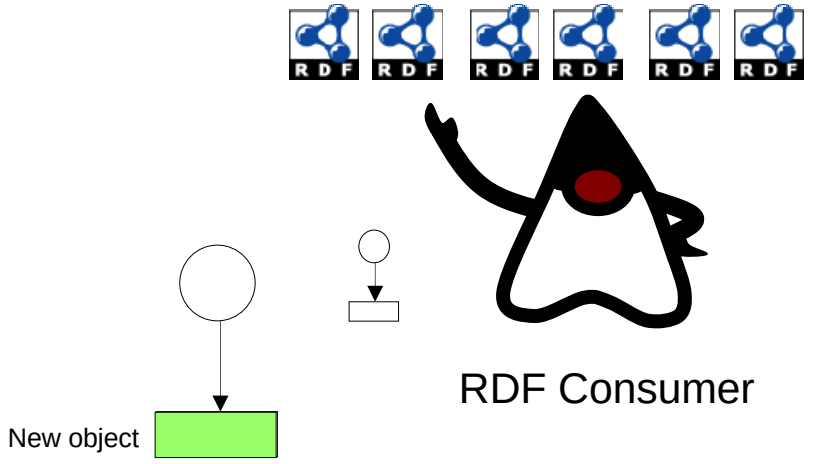
Model

Updating Model

ex6 demo

Using Statement's changeObject() method:

```
Resource jane = model.getResource("http://www.try.idv.tw/try#Jane");
Resource home =
                    model.getResource("http://www.try.idv.tw/try#Home");
Property locatedIn =
                  model.getProperty("http://www.try.idv.tw/try#locatedIn");

jane.getProperty(locatedIn).changeObject(home);
```



Removing a Statement from Model

Using Statement's remove() method:

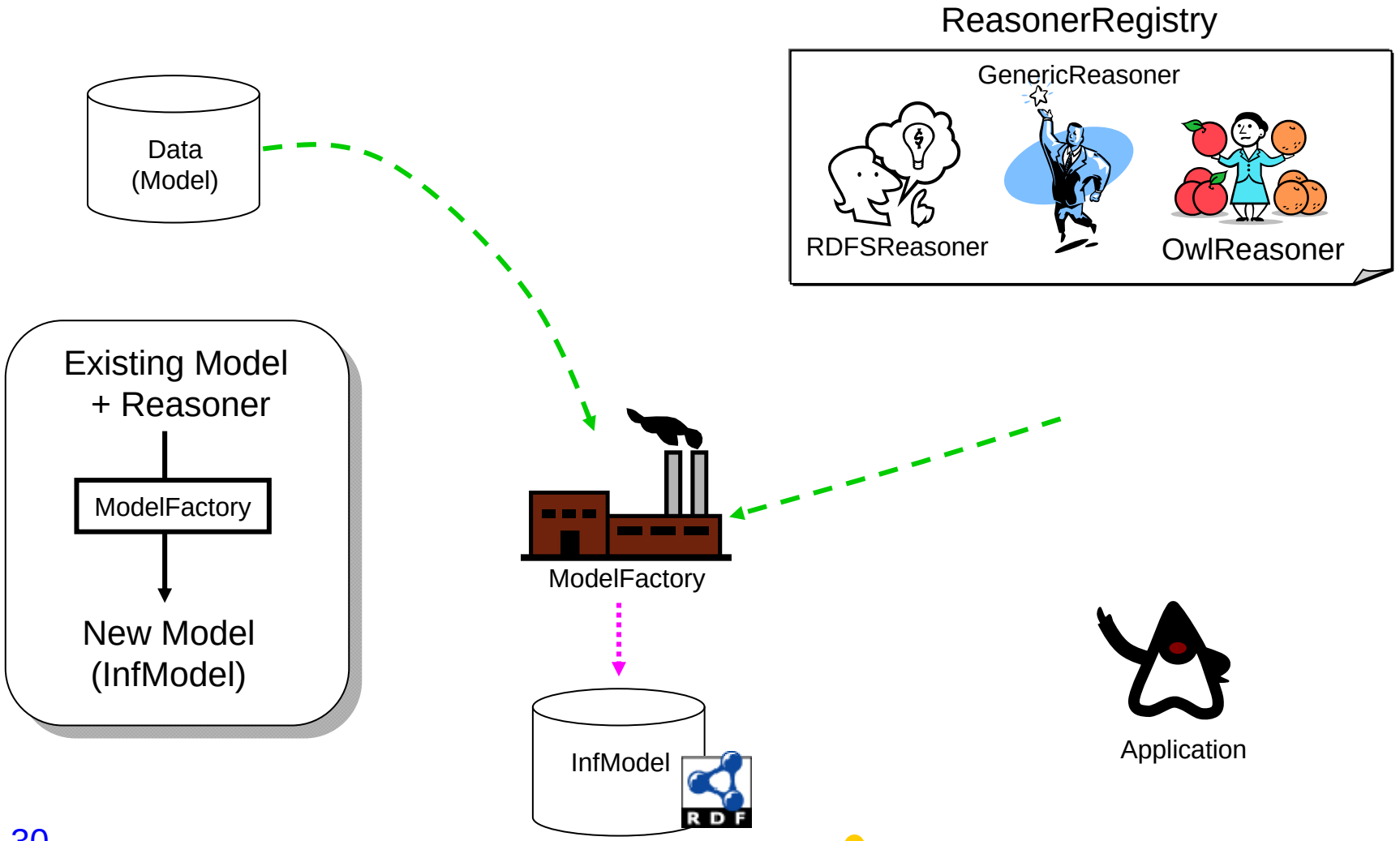
ex7 demo

```
Resource jane = model.getResource("http://www.try.idv.tw/try#Jane");
Property locatedIn =
                    model.getProperty("http://www.try.idv.tw/try#locatedIn");

jane.getProperty(locatedIn).remove;
```

Lab

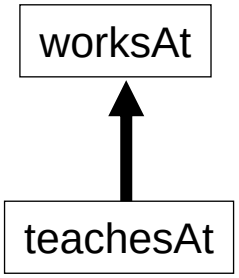
Basic Inference

Jena Inference Mechanism



A Simple RDFS Reasoner Example

```
Property teachesAtProp = data.createProperty(ns, "teachesAt");
Property worksAtProp = data.createProperty(ns, "worksAt");
data.add(teachesAtProp, RDFS.subPropertyOf, worksAtProp);
```




```
data.createResource(ns + "jane").addProperty(teachesAtProp, "NTU");
```

(Jane, teachesAt, NTU)

```
Reasoner reasoner = ReasonerRegistry.getRDFSReasoner();
InfModel infModel = ModelFactory.createInfModel(reasoner, data);
```

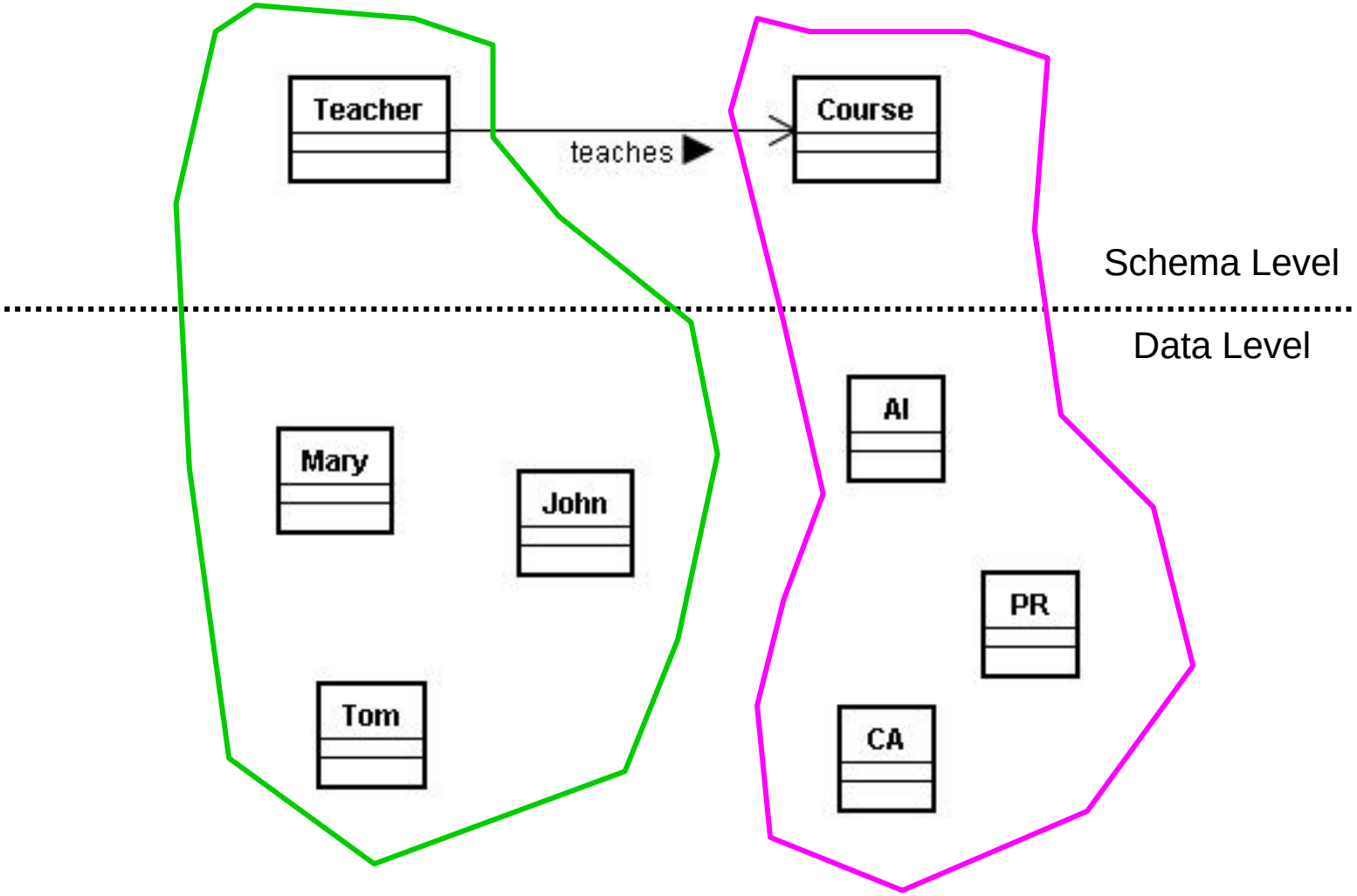
(Jane, worksAt, NTU)

RDFS Example demo

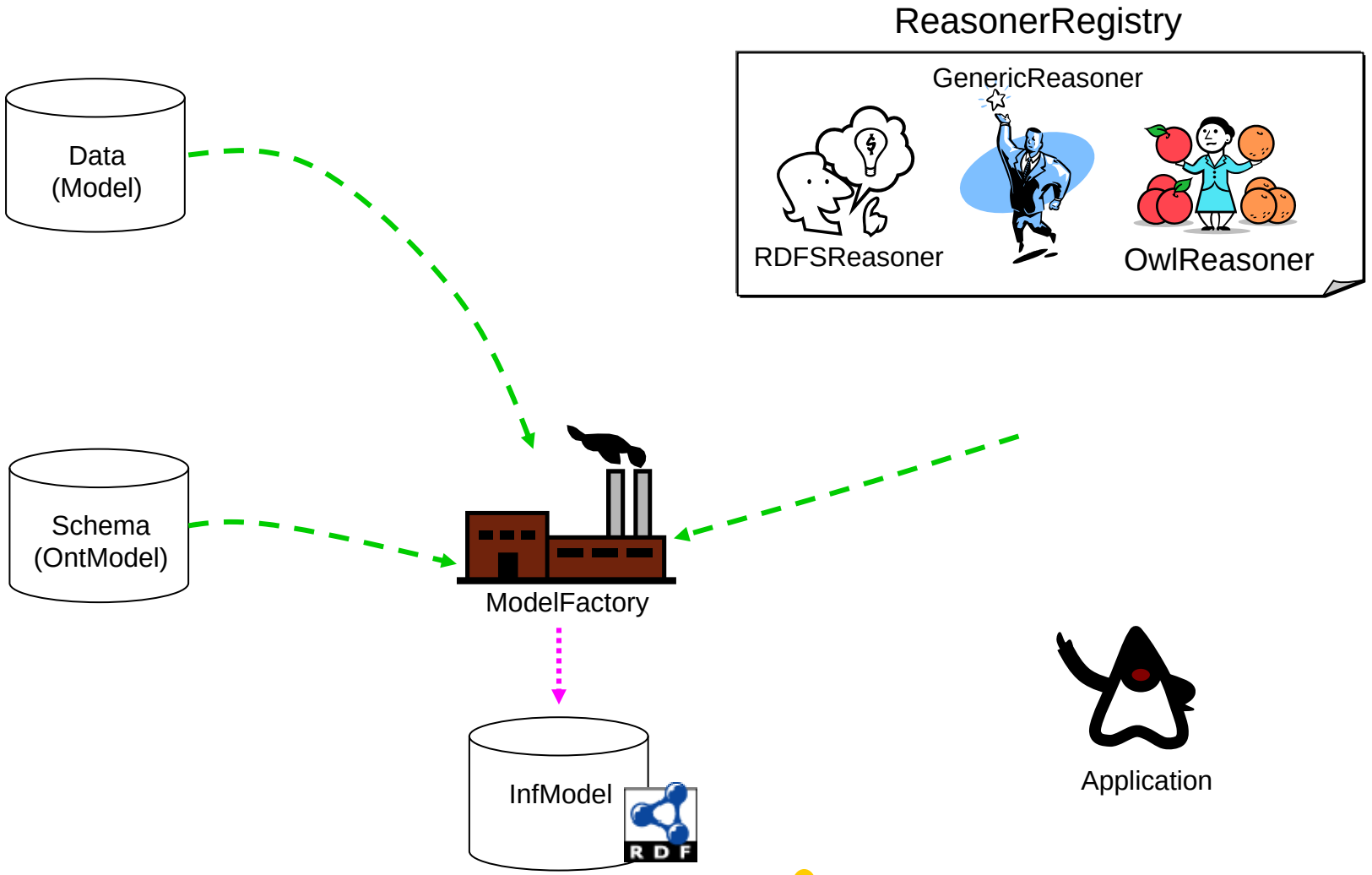
Lab

Ontological Inference

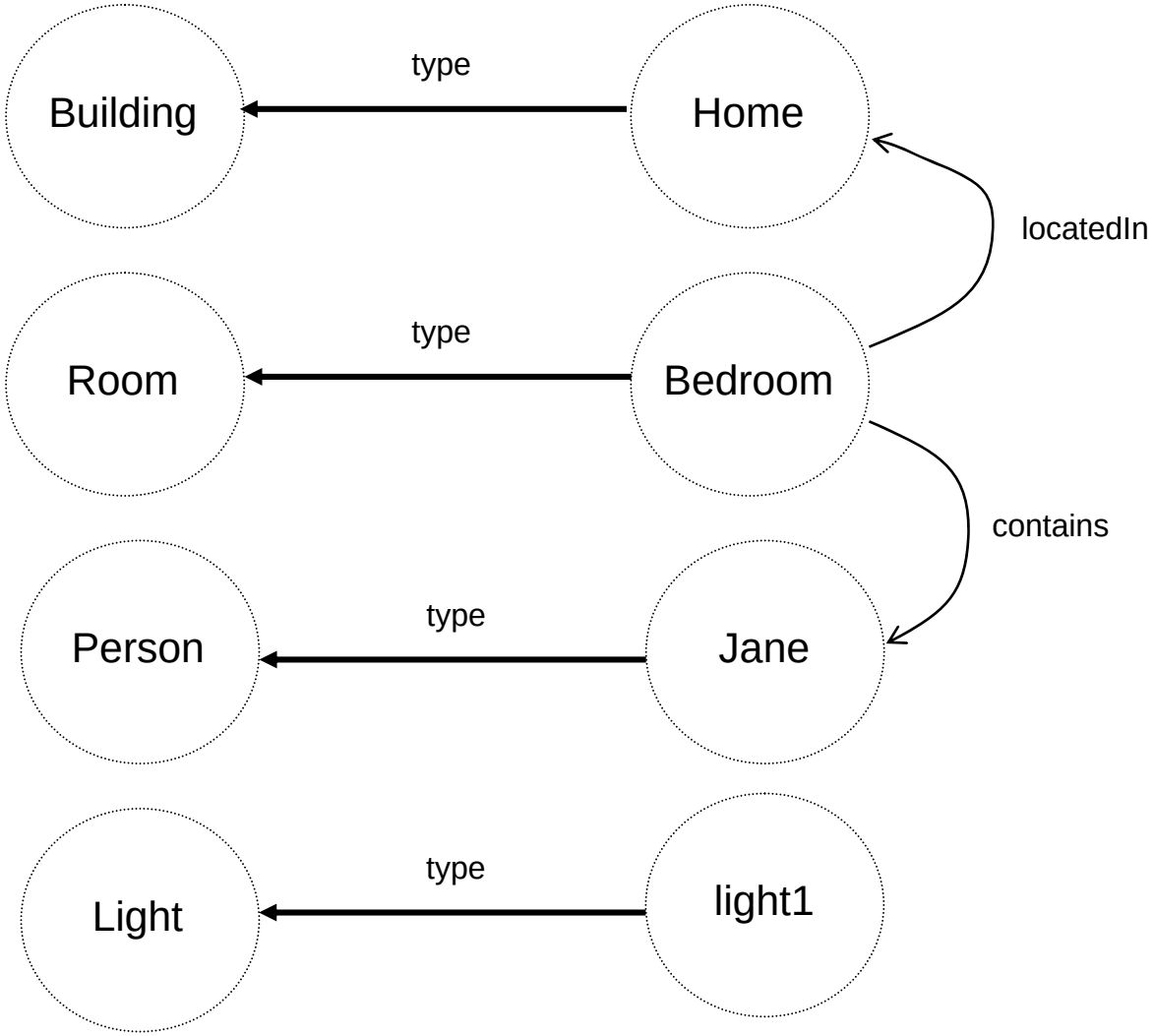
Modeling a Domain with Schema and Data

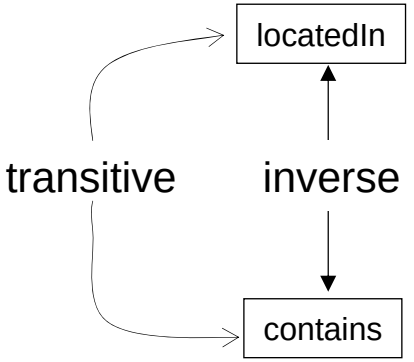


Separating Schema and Data



Example: The Smart Home Ontology





Creating OntModel with Ontology API

ex8 demo

```
OntModel m =
        ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM);
```

Model Configuration

```
m.createClass(ns + "Building");
m.createClass(ns + "Room");
m.createClass(ns + "Person");
m.createClass(ns + “Light");
```

Create 4 Class Definitions

```
ObjectProperty contains =
   m.createObjectProperty(ns + "contains", true).convertToTransitiveProperty();
ObjectProperty locatedIn =
   m.createObjectProperty(ns + "locatedIn").convertToTransitiveProperty();

contains.setInverseOf(locatedIn);
```

Inverse relationship

properties

A Ont Reasoner Example

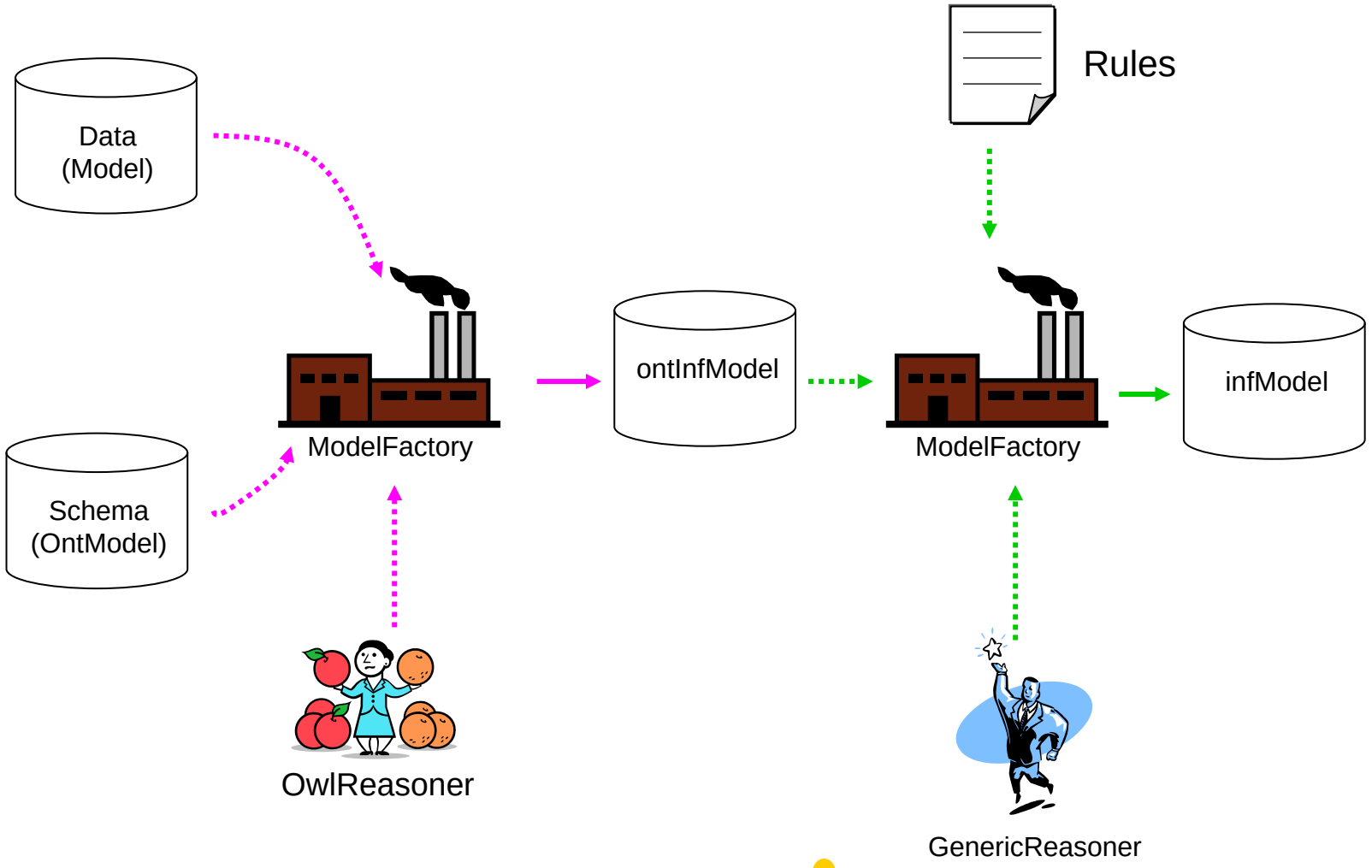
ex10 demo

```
OntModel schema = ModelFactory.createOntologyModel(…);
schema.read("file:./bin/advai/schema.owl");

Model data =
        FileManager.get().loadModel("file:./bin/advai/data.rdf");

Reasoner reasoner = ReasonerRegistry.getOWLReasoner();

InfModel infModel =
            ModelFactory.createInfModel(reasoner, schema, data);
```

Lab

Cascading Inference

Combining Generic and OWL Reasoners



An “Openlight” Rule

```
@prefix h: <http://www.ispace.tw/smarthome#> .
@include <RDFS>.
@include <OWL>.

[openlight:
    (?a h:locatedIn h:bedroom)
    (?a rdf:type h:Person)
    (?b rdf:type h:Light)
    (?b h:status 'off')  -> drop(3) (?b h:status 'on')
]
```

“if a person is in the bedroom then open all light”

Cascading Reasoners

```
OntModel schema =
      ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM_TRANS_INF);
schema.read("file:./bin/advai/inf/schema-inf-owl.owl");

Model data = FileManager.get().loadModel("file:./bin/advai/inf/data-inf-owl.rdf");

Reasoner owlReasoner = ReasonerRegistry.getOWLReasoner();
InfModel owlInfModel = ModelFactory.createInfModel(owlReasoner, schema, data);

GenericRuleReasoner reasoner =
        new GenericRuleReasoner(Rule.rulesFromURL("file:./bin/advai/inf/myrule.rule"));
reasoner.setDerivationLogging(true);

InfModel infModel = ModelFactory.createInfModel(reasoner, owlInfModel);
```

ex11 demo

Summary

- What we have learned
  - RDF / Model CRUD
  - OWL
  - OWL inference
  - OWL + Generic Rule engine inference
- Further readings
  - Jena Javadocs
  - Be sure to understand the meaning of advanced configuration mechanisms of models and reasoners.

Querying Model with SPARQL

```
Query query = …(Query String);
QueryExecution qexec = QueryExecutionFactory.create(query, model);

    try
    {
        ResultSet results = qexec.execSelect();
        for (; results.hasNext();)
        {
            QuerySolution soln = results.nextSolution();
            …// the solutions may be Resources or Literals
        }
    }
    finally
    {
        qexec.close();
    }
```

Sparql Demo

Persisting RDF Model

- Download required bundles
  - MySQL DB
  - MySQL JDBC Driver
- Install MySQL database
- Create jenadb
  - create database *jenatest* character set utf8 ;
- Persist with ModelMaker