Session II

Let's do some Semantic Web coding!

Exercise 1

- Create your FOAF file using the ttl syntax you learnt
- Link to your friends (or to people sitting next to you) using foaf:knows
- Upload it to your web space, and give yourself a URI!
- Ask us if you have any questions

Exercise 1 Solution

It should look something like this:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix : <http://dig.csail.mit.edu/2010/LinkedData/testdata/
carol.n3#> .

<> a foaf:PersonalProfileDocument;
     foaf:maker :i;
     foaf:primaryTopic :i.

    :i a foaf:Person;
         foaf:name "Carol Nobody";
         foaf:depiction <http://dig.csail.mit.edu/2008/02/rmp/
images/carol.jpg>;
         foaf:knows <http://dig.csail.mit.edu/2010/LinkedData/
testdata/bob#myself>.
```

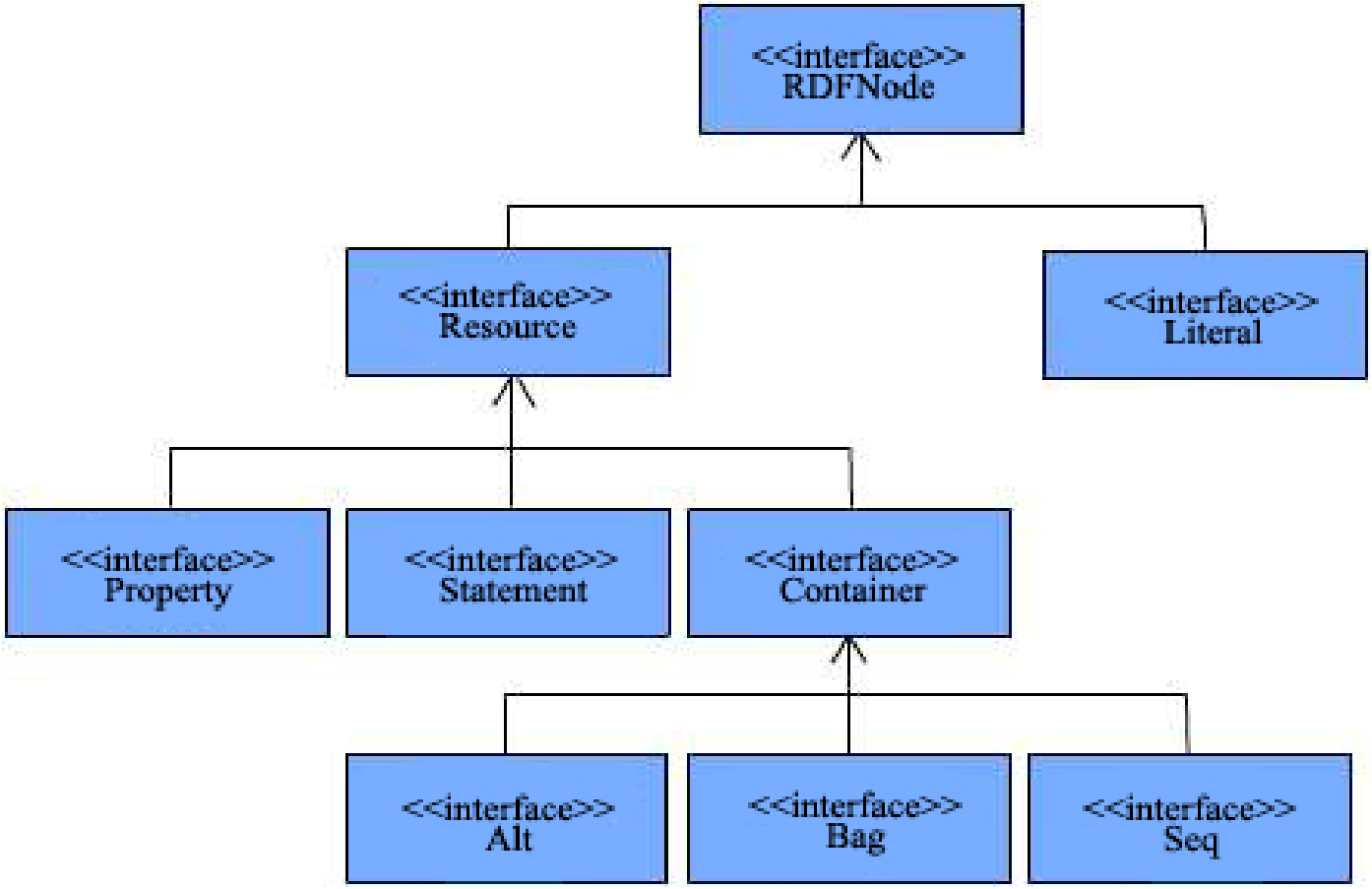
RDF in programming practice

- We will be using Java + Jena in our example
- Download Jena and set up your programming environment (e.g. Eclipse) *if you haven't already*

Jena Package

- jena.model
  - This is the key package for the application developer. It contains interfaces for model, resource, etc
- jena.mem
  - Contains an implementation of Jena API which stores all model state in main memory
- jena.common
  - Contains implementation classes

Jena API Structure



Jena Interfaces

Interface	What it represents in RDF
Model	Graph
Statement	Triple
Resource	A “thing” (with a URI)
Object	Resource or a Literal
Literal	Non nested “object”
Container	Special resource that represents a collection of things
Property	Property / Predicate

Programming in Jena

- RDF is parsed and results stored in the Model
  - The Model offers methods to retrieve:
    - triples
    - (property,object) pairs for a specific subject
    - (subject,property) pairs for specific object
  - The model can load several RDF files and merge the new statements automatically
  - The rest is conventional programming…
- Similar tools exist in Python, PHP, etc.
  (will be covered on Wednesday)

Exercise 2

- Create the FOAF file you created in Exercise 1 programatically using Jena.

Exercise 2 Solution

```
//Create the model
Model model = ModelFactory.createDefaultModel();

//Create all the resources
Resource friend = model.createResource(friendURI);

//Add proprties for the resources in a cascading style
Resource person = model.createResource(personURI)
                       .addProperty(RDF.type, FOAF.Person)
                       .addLiteral(FOAF.name, "Carol Nobody")
                       .addProperty(FOAF.knows, friend);

Resource foafDoc = model.createResource(documentURI)
                       .addProperty(RDF.type, FOAF.PersonalProfileDocument)
                       .addProperty(FOAF.maker, person)
                       .addProperty(FOAF.primaryTopic, person);;
```

Exercise 3

- Read your hosted FOAF file, and print out all your friends programatically using Jena.

Exercise 3 Solution

```
//Create the model
Model model = ModelFactory.createDefaultModel();

//Remember to give the "TTL" parameter. Default is "RDF/XML"
model.read("URI of your FOAF document", null, "N3");

NodeIterator it = model.listObjectsOfProperty(FOAF.knows);

while (it.hasNext()){
     RDFNode r = it.next();
     System.out.println(r.toString());
}
```

What is wrong here?

Exercise 3 Correct Solution

```
        Model model = ModelFactory.createDefaultModel();

        model.read("http://dig.csail.mit.edu/2010/LinkedData/testdata/
carol", "TTL");

        Resource carol = model.createResource("http://dig.csail.mit.edu/
2010/LinkedData/testdata/carol#i");

        //We are only interested in knowing who carol's friends are
        StmtIterator it = model.listStatements(carol, FOAF.knows, (RDFNode)
null);

        while (it.hasNext()){
            System.out.println(it.next().getObject().toString());
        }
```

Exercise 4

- Find common friends of ***Carol*** *(http://dig.csail.mit.edu/2010/LinkedData/testdata/carol)* and ***Eve*** *(http://dig.csail.mit.edu/2010/LinkedData/testdata/eve)*.

Exercise 4 Solution

```
        Model model = ModelFactory.createDefaultModel();

        model.read("http://dig.csail.mit.edu/2010/LinkedData/testdata/carol", "TTL");
        model.read("http://dig.csail.mit.edu/2010/LinkedData/testdata/eve", "TTL");

        Resource carol = model.createResource("http://dig.csail.mit.edu/2010/
LinkedData/testdata/carol#i");
        Resource eve = model.createResource("http://dig.csail.mit.edu/2010/
LinkedData/testdata/eve#i");

        List<Statement> carolFriends = model.listStatements(carol, FOAF.knows,
(RDFNode) null).toList();
        List<Statement> eveFriends = model.listStatements(eve, FOAF.knows, (RDFNode)
null).toList();

        for (Statement c : carolFriends){
            for (Statement e : eveFriends){
                if (c.getObject().equals(e.getObject())){
                    System.out.println(c.getObject().toString());
                }
            }
        }
```

References

- Jena documentation: http://jena.sourceforge.net/documentation.html