SETUP FOR A GENERIC LINKED DATA PLATFORM

CHRISTOPHE DEBRUYNE, ADAPT @ TRINITY COLLEGE DUBLIN

INTRODUCTION

This document aims to provide the reader with a step-by-step tutorial for creating a generic Linked Data platform from the uplift process – generating RDF from non-RDF sources – to connecting a Linked Data frontend with content negotiation.

CHOOSING AN APPROPRIATE URI

Remember that one needs to provide HTTP URIs to retrieve useful information. So the first step is to think about a domain that will serve as the base URI for your resource identifiers. For this tutorial, we will assume that the base URI for all resources will be http://data.example.org/.

ANNOTATING A RELATIONAL DATABASE WITH R2RML

We will annotate a relational database with R2RML[1] to generate RDF with an appropriate processor. First, we assume the existence of a database "tutorial" with two tables. A MySQL dump file for this table can be found in Appendix 1.

Looking at the examples provided by the R2RML W3C Recommendation, create a mapping that

- Uses FOAF (you can use ns "foaf": http://xmlns.com/foaf/0.1/)
- Uses geo (you can use ns "geo" for (http://www.w3.org/2003/01/geo/wgs84_pos#)
- Declares a TriplesMap for the table "Place"
    - Each record is an instance of geo:SpatialThing
    - Each record has a foaf:name and an rdfs:label
- Declares a TriplesMap for the table Person
    - Each record is an instance of foaf:Person
    - Each record has a given name, family name and a name (the concatenation of both). Use the name for the rdfs:label. Note that the family name can be NULL. How would you solve this?
- Relate people with places with the predicate foaf:based_near.

Use a R2RML Processor to generate triples. Many implementations exist. For this tutorial, db2triples[2] or ADAPT's implementation[3] suffice. Please feel free to try the latter and report me any problems you might have! ☺

[1] https://www.w3.org/TR/r2rml/

The implementations are straightforward. With db2triples, you can execute the mapping with:

```
$ java -jar db2triples.jar -b tutorial -m r2rml -u XXX -p XXX -r mapping1.ttl
```

The implementation I have developed uses a configuration file and provides support for named graphs, which is not the case for db2triples. A configuration file could look as follows:

```
connectionURL = jdbc:mysql://localhost/tutorial
user = XXX
password = XXX
mappingFile = mapping.ttl
outputFile = output.ttl
format = TURTLE
```

And the RDF is generated with the following command:

```
$ java -jar r2rml-0.0.1-SNAPSHOT.jar config.properties
```

Though I urge you to try to create your own mapping, the examples in the recommendation should be sufficient, a mapping is provided in Appendix 2. The RDF generated with that mapping is provided in Appendix 3.

SETTING UP THE TRIPLESTORE AND SPARQL ENDPOINT

Setting up a triplestore and SPARQL endpoint is straightforward. May suites exist:

- Stardog – http://stardog.com/ (free for non-commercial purposes)
- Virtuoso – https://github.com/openlink/virtuoso-opensource
- Parliament – http://parliament.semwebcentral.org/
- Apache Jena and Fuseki – https://jena.apache.org
- …

For this tutorial, we are going to use Apache Jena and Fuseki. If you wish to quickly try out a SPARQL endpoint, you can start an in-memory dataset that allows for updating as follows:

```
$ ./fuseki-server --update --mem /ds
```

Fuseki runs by default on localhost:3030. You can now add the triples manually in the dataset with the interface provided (see below).

---

[2] https://github.com/antidot/db2triples
[3] https://github.com/CNGL-repo/r2rml

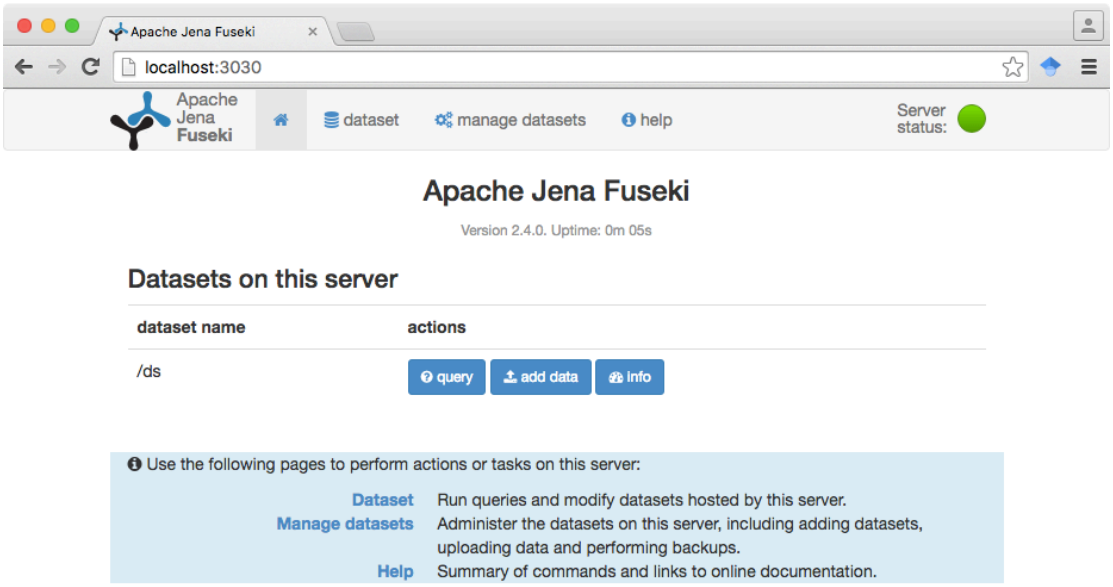


Figure 1 Click on "add data" to manually load triples in the dataset "ds" that we have created in the terminal.

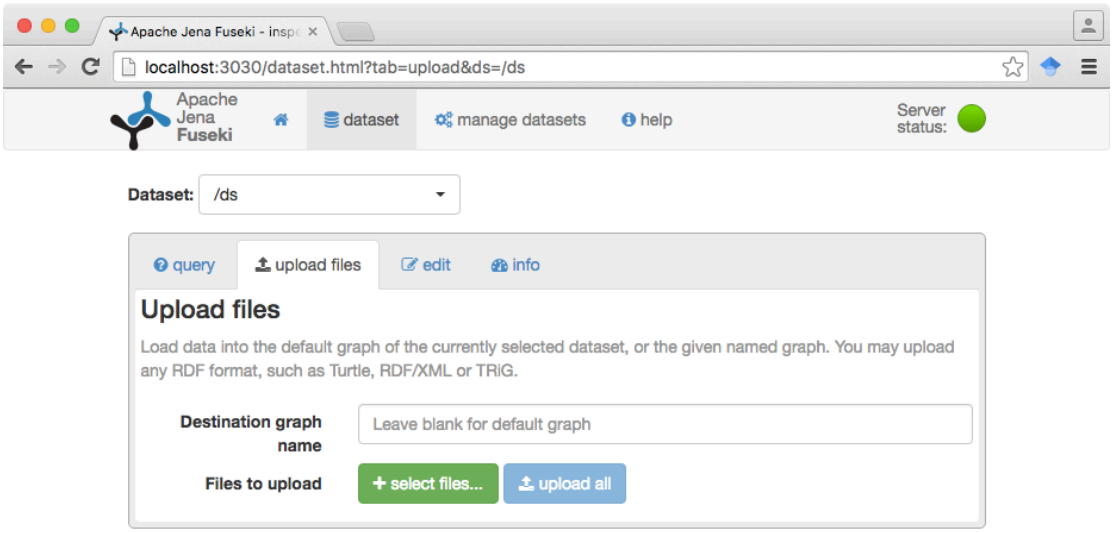


Figure 2 You can upload one or more files and even specify the graphs in which the triples have to be loaded.

With the triples loaded, you can now go to the next section to set up the Linked Data frontend. Of course, this is an in-memory dataset and all data will be lost once the server is shutdown. For this tutorial, we are going to create a triplestore with Jena TDB and run Jena Fuseki in such a way to communicate with that triplestore. For this tutorial, we are going to create and configure a read-only triplestore. More information to configure endpoint with support for updates and reasoning can be found in the documentation.

Creating a triplestore called tutorialDB with tdbloader2 (in the "bin" directory of Apache Jena) is done as follows:

```
$ apache-jena-3.1.0/bin/tdbloader2 --loc tutorialDB output.ttl
```

WATCH OUT! The Jena tdbloaders does not like spaces in paths.

Note that many files can be given, but we only have one outputfile. This results in a directory called "tutorialDB" containing the triples and indices. Now we need to configure Jena Fuseki to use tutorialDB as a dataset. This can be done with the following command:

```
./fuseki-server --loc=tutorialDB /ds
```

Note that Jena Fuseki also provides means to configure services with a configuration file and that its GUI also allows one to manage (new) datasets.

SETTING UP THE LINKED DATA FRONTEND

There are a few Linked Data frontends available. Most of them serve fairly generic HTML view to end-users. This is why it is sometimes worthwhile to build your frontend with a combination of existing services (e.g., web pages), reverse proxies and URL rewrite rules. For this tutorial, however, we will setup a simple frontend with Pubby[4]. Pubby provides content negotiation and simple HTML views for RDF resources. Pubby runs easily in a container such as Jetty[5].

First we need to setup our system such that it resolves http://data.example.org/ to localhost. On *NIX systems, you can change the /etc/hosts file to include the following.

```
127.0.0.1               data.example.org
```

Download Pubby and build the project with Maven. Some tests might fail and – depending on your version of Java – the generation of documentation as will. You can package the project with the following command.

```
$ mvn package -DskipTests -Dmaven.javadoc.skip=true
```

In the "target" directory you will find the "pubby" directory that has been created. Rather than the war – which will get expanded in an application container – we will use that directory as we will need to change the configuration file.

Download Jetty and copy the directory "pubby" into the "webapps" directory of Jetty. Rename "pubby" to "ROOT", as we want to have the service running from http://data.example.org/ instead of http://data.example.org/pubby/.

Using Pubby's documentation and the examples provided, configure Pubby to communicate with the SPARQL endpoint that you have set up in the previous section. The configuration file config.ttl may look as follows:

```
# Prefix declarations to be used in RDF output
@prefix conf: <http://richard.cyganiak.de/2007/pubby/config.rdf#> .
@prefix ont: <http://www.geohive.ie/ontology/osi#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .

# Server configuration section
<> a conf:Configuration;
    conf:projectName "Linked Data Tutorial";
    conf:projectHomepage <http://www.example.org/>;
    conf:webBase <http://data.example.org/>;
    conf:usePrefixesFrom <>;
    conf:defaultLanguage "en";
```

[4] http://wifo5-03.informatik.uni-mannheim.de/pubby/
[5] http://www.eclipse.org/jetty/

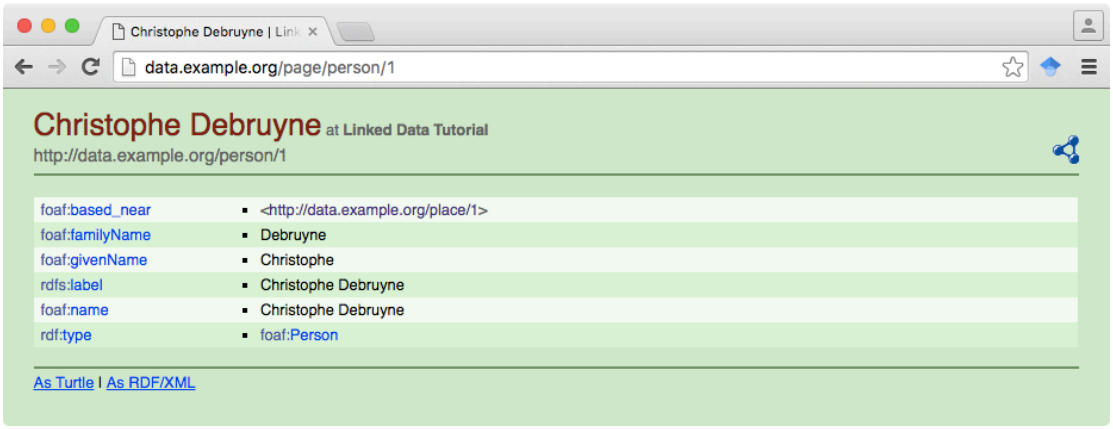
```
    conf:dataset [
        conf:sparqlEndpoint <http://localhost:3030/ds/sparql>;
        conf:datasetBase <http://data.example.org/> ;
        conf:fixUnescapedCharacters "(),'!$&*+;=@";
               conf:resourceDescriptionQuery "DESCRIBE ?__this__";
     ];
     .
```

Start the Jetty web server. Make sure that you have sufficient permissions to use port 80 (the default HTTP port).

```
$ sudo java -jar start.jar
```

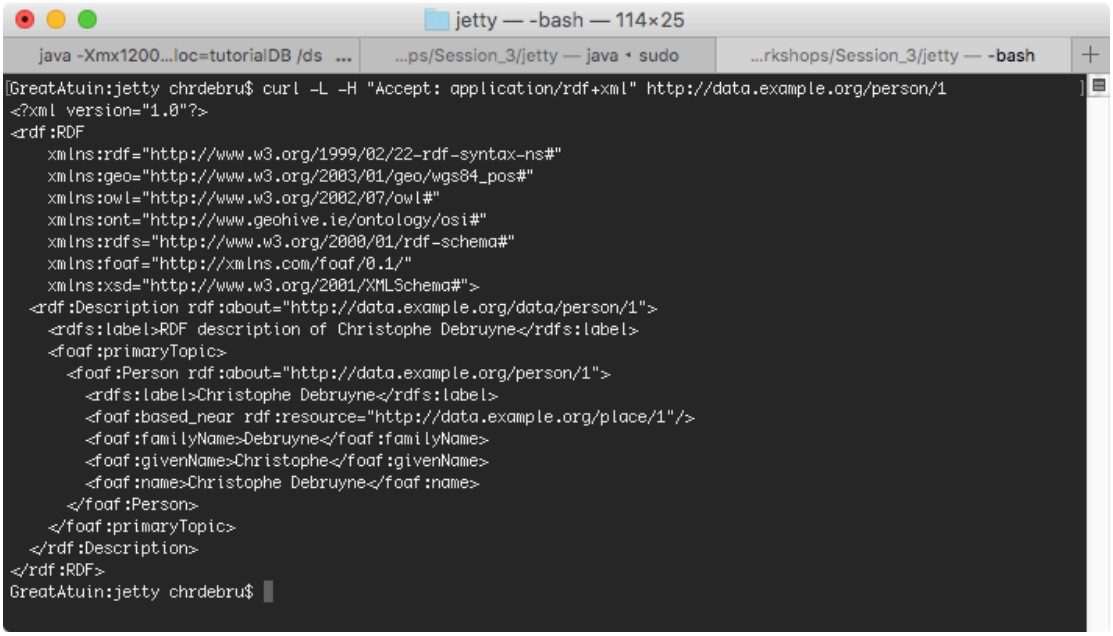
TESTING THE LINKED DATA FRONTEND

When one puts http://data.example.org/person/1 in the browser, you see that one is redirected to http://data.example.org/page/person/1.



To obtain RDF, we can execute the following command in the terminal.

```
$ curl -L -H "Accept: application/rdf+xml" http://data.example.org/person/1
```



APPENDIX 1: MYSQL DATABASE DUMP

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";
CREATE TABLE `Person` (
`id` int(11) NOT NULL,
  `fname` varchar(30) DEFAULT NULL,
  `lname` varchar(30) DEFAULT NULL,
  `place` int(11) DEFAULT NULL
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1;
INSERT INTO `Person` (`id`, `fname`, `lname`, `place`) VALUES
(1, 'Christophe', 'Debruyne', 1),
(2, 'Victor', NULL, 2),
(3, 'Louis', NULL, 2),
(4, 'Bettina', NULL, 2),
(5, 'Kevin', 'Debruyne', NULL),
(6, 'Robert', 'Meersman', 6);
CREATE TABLE `Place` (
`id` int(11) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=latin1;
INSERT INTO `Place` (`id`, `name`) VALUES
(1, 'Dublin'),
(2, 'Brussels'),
(5, 'Ghent'),
(6, 'Vienna');
ALTER TABLE `Person`
 ADD PRIMARY KEY (`id`), ADD KEY `place` (`place`);
ALTER TABLE `Place`
 ADD PRIMARY KEY (`id`);
ALTER TABLE `Person`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=7;
ALTER TABLE `Place`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=9;
ALTER TABLE `Person`
ADD CONSTRAINT `fk_place` FOREIGN KEY (`place`) REFERENCES `Place` (`id`);
```

APPENDIX 2: EXAMPLE OF THE MAPPING FILE

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<PlaceTM>
  a rr:TriplesMap ;

  rr:logicalTable [ rr:tableName "Place" ] ;

  rr:subjectMap [
    rr:template "http://data.example.org/place/{id}" ;
    rr:class geo:SpatialThing ;
  ] ;

  rr:predicateObjectMap [
    rr:predicate foaf:name, rdfs:label ;
    rr:objectMap [ rr:column "name" ] ;
  ] ;
  .

<PersonTM>
  a rr:TriplesMap;

  rr:logicalTable [
    rr:sqlQuery """
      SELECT *, IF(lname IS NULL, fname, CONCAT(fname, ' ', lname)) as name
      FROM Person WHERE 1
    """
  ] ;

  rr:subjectMap [
    rr:template "http://data.example.org/person/{id}" ;
    rr:class foaf:Person ;
  ] ;

  rr:predicateObjectMap [
    rr:predicate foaf:givenName ;
    rr:objectMap [ rr:column "fname" ] ;
  ] ;

  rr:predicateObjectMap [
    rr:predicate foaf:familyName ;
    rr:objectMap [ rr:column "lname" ] ;
  ] ;

  rr:predicateObjectMap [
    rr:predicate foaf:name ;
    rr:objectMap [ rr:column "name" ] ;
  ] ;

  rr:predicateObjectMap [
    rr:predicate foaf:based_near ;
    rr:objectMap [
      rr:parentTriplesMap <PlaceTM> ;
      rr:joinCondition [
        rr:child "place" ;
        rr:parent "id"
      ] ;
    ] ;
  ] ;
  .
```

APPENDIX 3: GENERATED RDF

```
<http://data.example.org/place/6>
        a       <http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Vienna" ;
        <http://xmlns.com/foaf/0.1/name>
                "Vienna" .

<http://data.example.org/place/1>
        a       <http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Dublin" ;
        <http://xmlns.com/foaf/0.1/name>
                "Dublin" .

<http://data.example.org/person/5>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Kevin Debruyne" ;
        <http://xmlns.com/foaf/0.1/familyName>
                "Debruyne" ;
        <http://xmlns.com/foaf/0.1/givenName>
                "Kevin" ;
        <http://xmlns.com/foaf/0.1/name>
                "Kevin Debruyne" .

<http://data.example.org/person/3>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Louis" ;
        <http://xmlns.com/foaf/0.1/based_near>
                <http://data.example.org/place/2> ;
        <http://xmlns.com/foaf/0.1/givenName>
                "Louis" ;
        <http://xmlns.com/foaf/0.1/name>
                "Louis" .

<http://data.example.org/place/2>
        a       <http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Brussels" ;
        <http://xmlns.com/foaf/0.1/name>
                "Brussels" .

<http://data.example.org/person/1>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Christophe Debruyne" ;
        <http://xmlns.com/foaf/0.1/based_near>
                <http://data.example.org/place/1> ;
        <http://xmlns.com/foaf/0.1/familyName>
                "Debruyne" ;
        <http://xmlns.com/foaf/0.1/givenName>
                "Christophe" ;
        <http://xmlns.com/foaf/0.1/name>
                "Christophe Debruyne" .

<http://data.example.org/person/6>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Robert Meersman" ;
        <http://xmlns.com/foaf/0.1/based_near>
                <http://data.example.org/place/6> ;
        <http://xmlns.com/foaf/0.1/familyName>
                "Meersman" ;
        <http://xmlns.com/foaf/0.1/givenName>
                "Robert" ;
        <http://xmlns.com/foaf/0.1/name>
                "Robert Meersman" .

<http://data.example.org/place/5>
        a       <http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing> ;
```

```
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Ghent" ;
        <http://xmlns.com/foaf/0.1/name>
                "Ghent" .

<http://data.example.org/person/4>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Bettina" ;
        <http://xmlns.com/foaf/0.1/based_near>
                <http://data.example.org/place/2> ;
        <http://xmlns.com/foaf/0.1/givenName>
                "Bettina" ;
        <http://xmlns.com/foaf/0.1/name>
                "Bettina" .

<http://data.example.org/person/2>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "Victor" ;
        <http://xmlns.com/foaf/0.1/based_near>
                <http://data.example.org/place/2> ;
        <http://xmlns.com/foaf/0.1/givenName>
                "Victor" ;
        <http://xmlns.com/foaf/0.1/name>
                "Victor" .
```