RDF and RDB 2

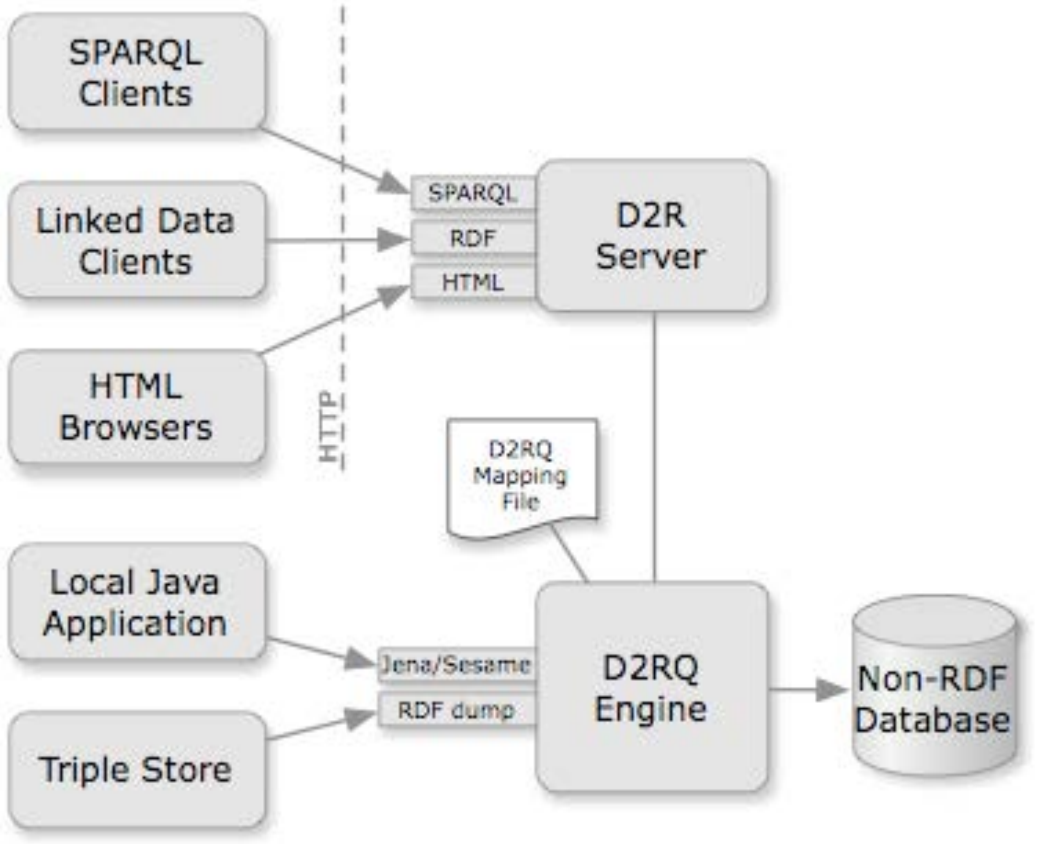
D2RQ

D2RQ showed the way

- Early system to expose relational data as RDF
  - See http://d2rq.org/
  - Open source: https://github.com/d2rq/d2rq
  - Still widely used
- Lets you
  - Query a non-RDF database using SPARQL
  - Access database content as linked data over Web
  - Dump database content in RDF formats
  - Access non-RDF database using Apache Jena API

D2RQ

- *D2RQ mapping language file* describes relation between ontology & RDB
- *D2R server* provides HTML & linked data views & SPARQL endpoint
- *D2RQ engine* uses map-pings to rewrite Jena & Sesame API calls to SQL queries & generates RDF dumps in various formats



D2RQ Features

- Browsing database contents: Web interface for navigation through the RDF contents for people
- Resolvable URIs: D2R Server assigns a resolvable URI to each entity in the database
- Content negotiation: HTML & RDF versions share URIs; HTTP content negotiation fixes version
- SPARQL: Both an endpoint and explorer provided
- BLOBs and CLOBs: Support for serving up values as files (e.g., PDFs, images)
- Not surprisingly, no inferencing

D2RQ Mapping Language

- The mapping is defined in RDF
- D2RQ generates a default map using a standard heuristic:
  - Each DB **table** has infor. about one **type of thing**
  - Each table **row** represents **one object**
  - First column is **key** => defines the object
  - Other columns represent **properties**
- Edit default mapping or create your own

Let’s do it

- Need: relational DBMS, Java, Web server
- Clone or download D2RQ git repo
- Compile with: *ant jar*
  - Install java and ant as needed
- Create default mapping from a database
- Start D2RQ server on a port
  - Send it SPARQL queries
  - Access it via html

A simple database

Load lab.sql into mysql

```
mysql –u demo –p demo
...
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql> source lab.sql
...
```

lab.sql is an sql dump file

```
DROP SCHEMA IF EXISTS lab;
CREATE SCHEMA lab;
USE lab;

Drop TABLE IF EXISTS people;

CREATE TABLE people (
  `Name` varchar(50),
  `Age` INT default NULL,
  `Mobile` varchar(50) default NULL,
  PRIMARY KEY  (`Name`)
);

INSERT INTO people (`Name`, `Age`,
`Mobile`) VALUES
('Al Turing', 32, '443-253-3863'),
('Don Knuth', 25, '410-228-6282'),
('Chuck Babbage', 38, '410-499-1282');
```

A simple database

```
mysql> use lab; show tables;
+----------------+
| Tables_in_lab |
+----------------+
| people        |
+----------------+
mysql> desc people;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Name   | varchar(50) | NO   | PRI |         |       |
| Age    | int(11)     | YES  |     | NULL    |       |
| Mobile | varchar(50) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
mysql> select * from people;
+---------------+------+--------------+
| Name          | Age  | Mobile       |
+---------------+------+--------------+
| Al Turing     |   32 | 443-253-3863 |
| Don Knuth     |   25 | 410-228-6282 |
| Chuck Babbage |   38 | 410-499-1282 |
+---------------+------+--------------+
```

The default model

- The *people table* has info of things of type people <*http://ebiq.org/o/labvocab/resource/people*>
- Each row in the table has information about one instance of a person
- The first column is the key and is used both
  - As the identifier for a person instance <*http://localhost/people/Chuck_Babbage*>
  - For the rdf:label for a person instance
- Properties of a person are: name, age & mobile <http://ebiq.org/o/labvocab/resource/people_Age>

Generating RDF mappings

- D2RQ generates a **default mapping** directly from the database

```
% d2rq/generate-mapping –u demo –w3c  \
     -o  lab_map.ttl  jdbc:mysql://127.0.0.1/lab
```

  - -u arg: user for database access
  - -o arg: file to write mapping to
  - --w3c flag: use W3C compatible mapping format
  - Last arg: string JDBC uses to access database table
- Resulting mapping can be edited as desired

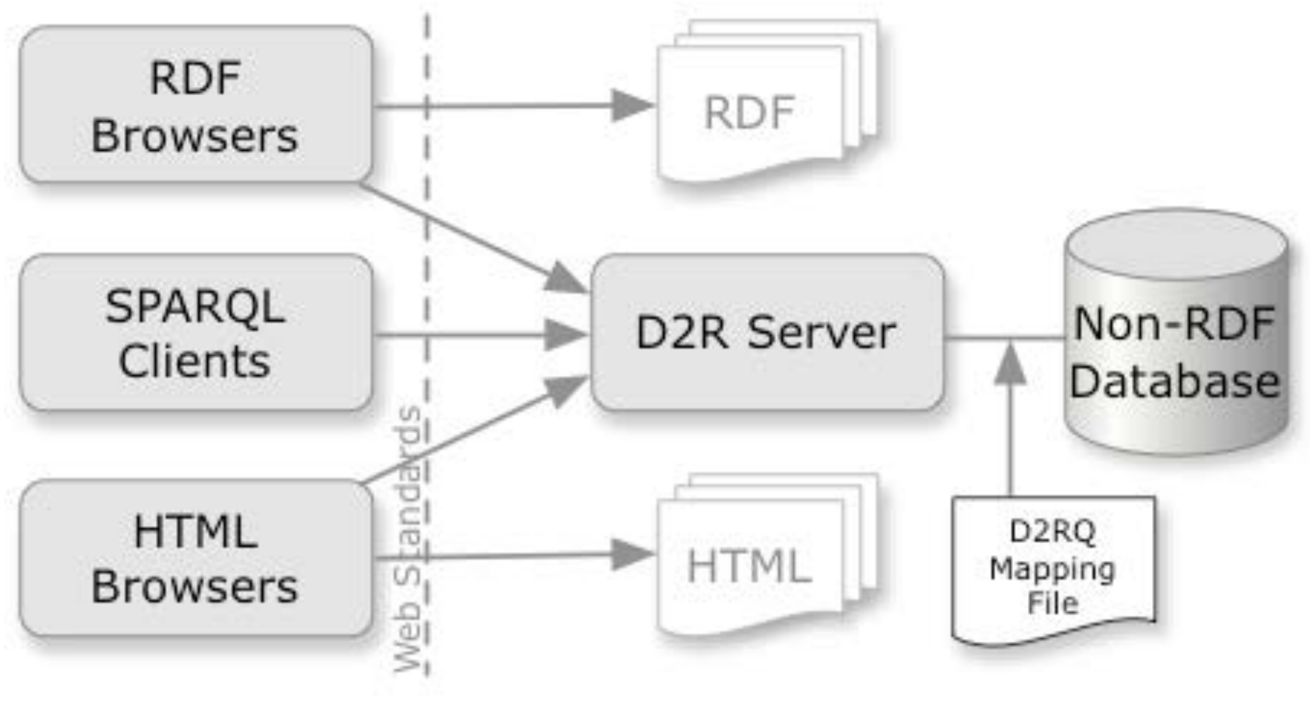
The Default D2RQ mapping

```
@prefix ...
Map:database a d2rq:Database;
  d2rq:jdbcDriver "com.mysql.jdbc.Driver";
  d2rq:jdbcDSN "jdbc:mysql://127.0.0.1/lab";
  d2rq:username "demo”;
  jdbc:autoReconnect "true";
  jdbc:zeroDateTimeBehavior "convertToNull”; .
map:people a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern "people/@@people.Name|
urlify@@";
  d2rq:class vocab:people;
  d2rq:classDefinitionLabel "people”; .
map:people__label a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:people;
  d2rq:property rdfs:label;
  d2rq:pattern "people #@@people.Name@@”;.
```

```
map:people_Name a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:people;
  d2rq:property vocab:people_Name;
  d2rq:propertyDefinitionLabel "people Name";
  d2rq:column "people.Name"; .
map:people_Age a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:people;
  d2rq:property vocab:people_Age;
  d2rq:propertyDefinitionLabel "people Age";
  d2rq:column "people.Age";
  d2rq:datatype xsd:int; .
map:people_Mobile a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:people;
  d2rq:property vocab:people_Mobile;
  d2rq:propertyDefinitionLabel "people Mobile";
  d2rq:column "people.Mobile"; .
```

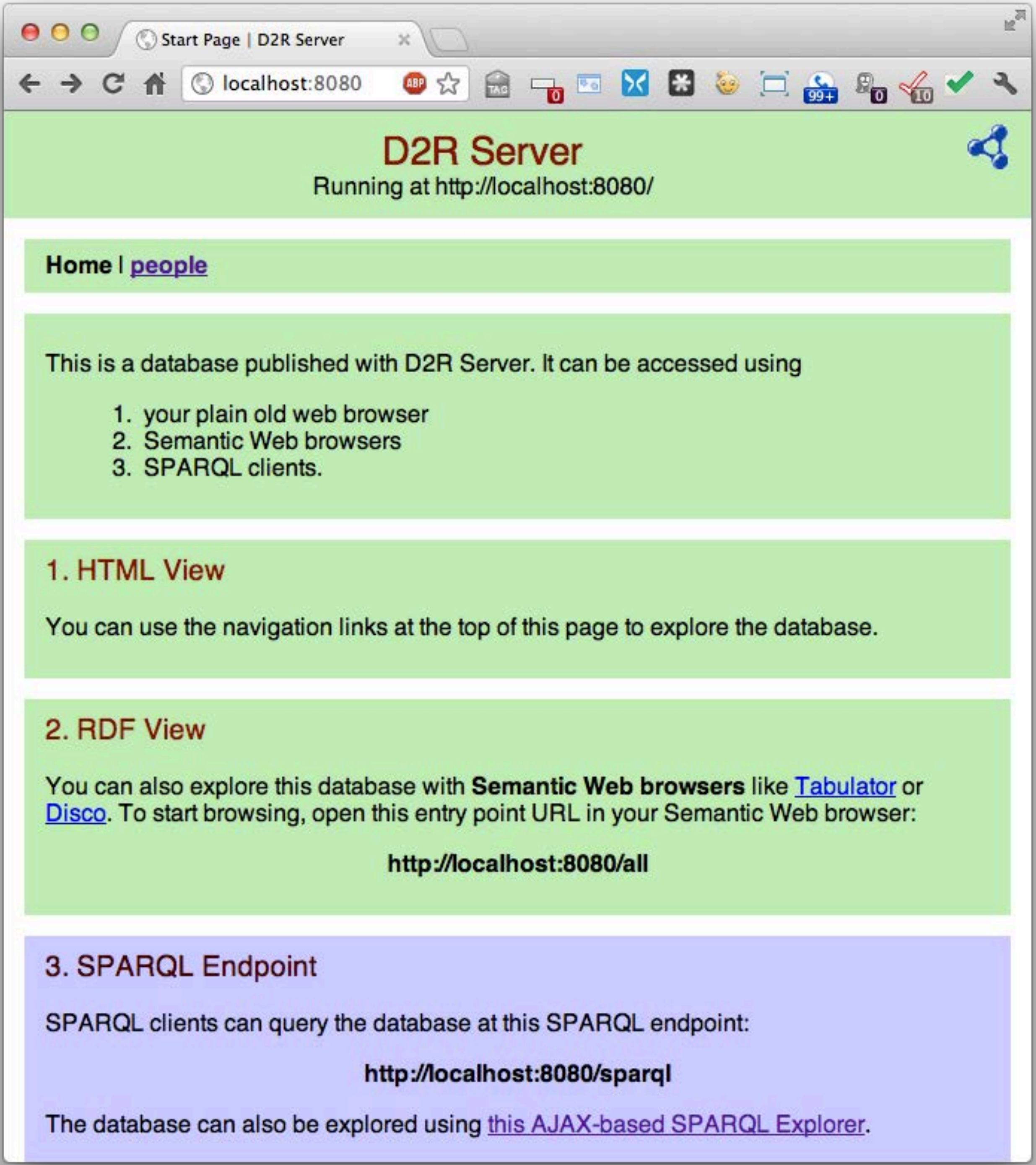
D2r Server

- The d2r-server provides real-time access to rdf data via several protocols
  - d2r-server -port 8081 ../lab_map.ttl



Access via D2R server

- Explore via HTML
- Via SPARQL endpoint



D2R Server
Running at http://localhost:8080/

Home | people

This is a database published with D2R Server. It can be accessed using

1. your plain old web browser
2. Semantic Web browsers
3. SPARQL clients.

1. HTML View

You can use the navigation links at the top of this page to explore the database.

2. RDF View

You can also explore this database with **Semantic Web browsers** like Tabulator or Disco. To start browsing, open this entry point URL in your Semantic Web browser:

http://localhost:8080/all

3. SPARQL Endpoint

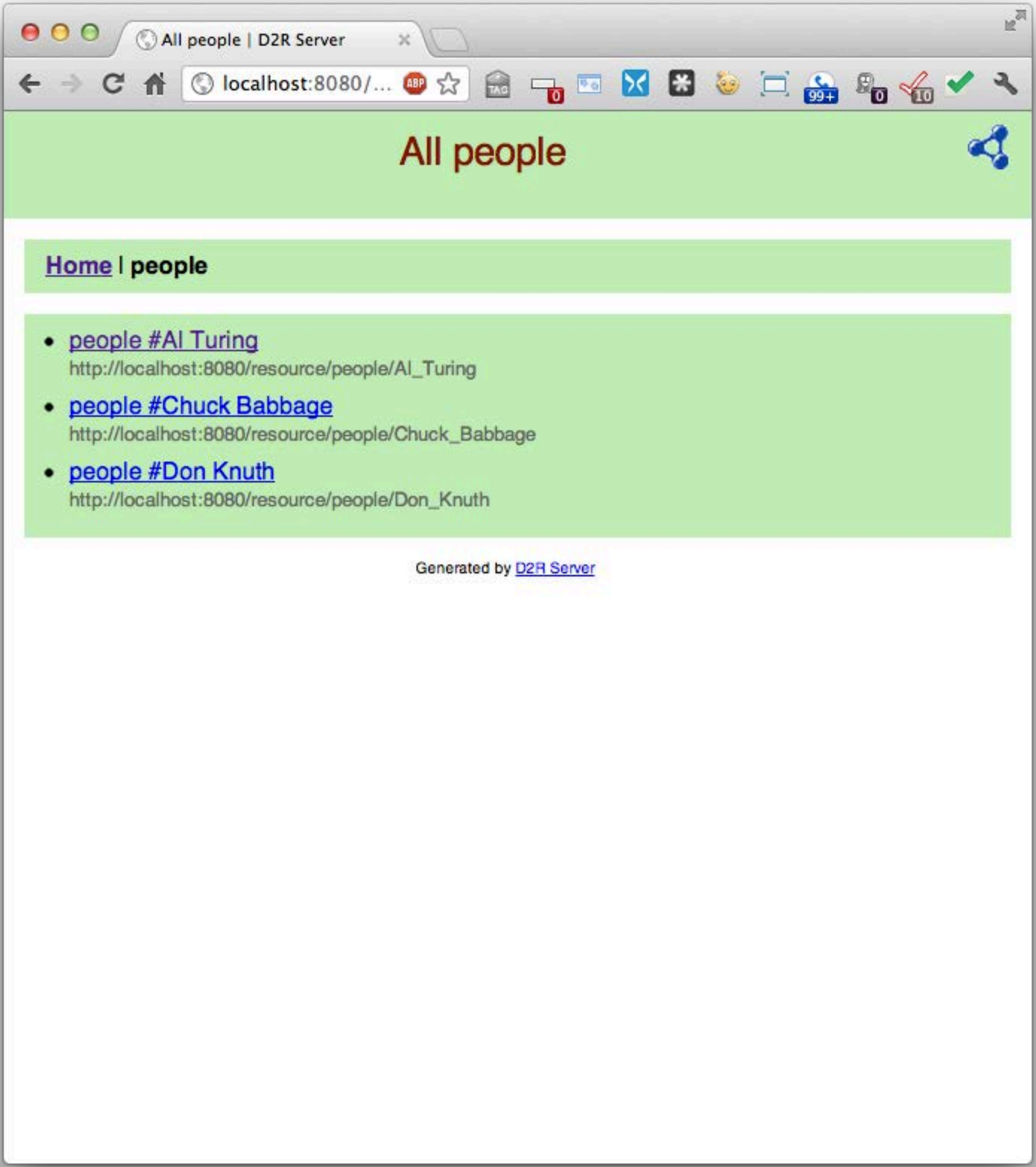
SPARQL clients can query the database at this SPARQL endpoint:

http://localhost:8080/sparql

The database can also be explored using this AJAX-based SPARQL Explorer.

Access via D2R server

- Explore via HTML
- Via SPARQL endpoint



All people

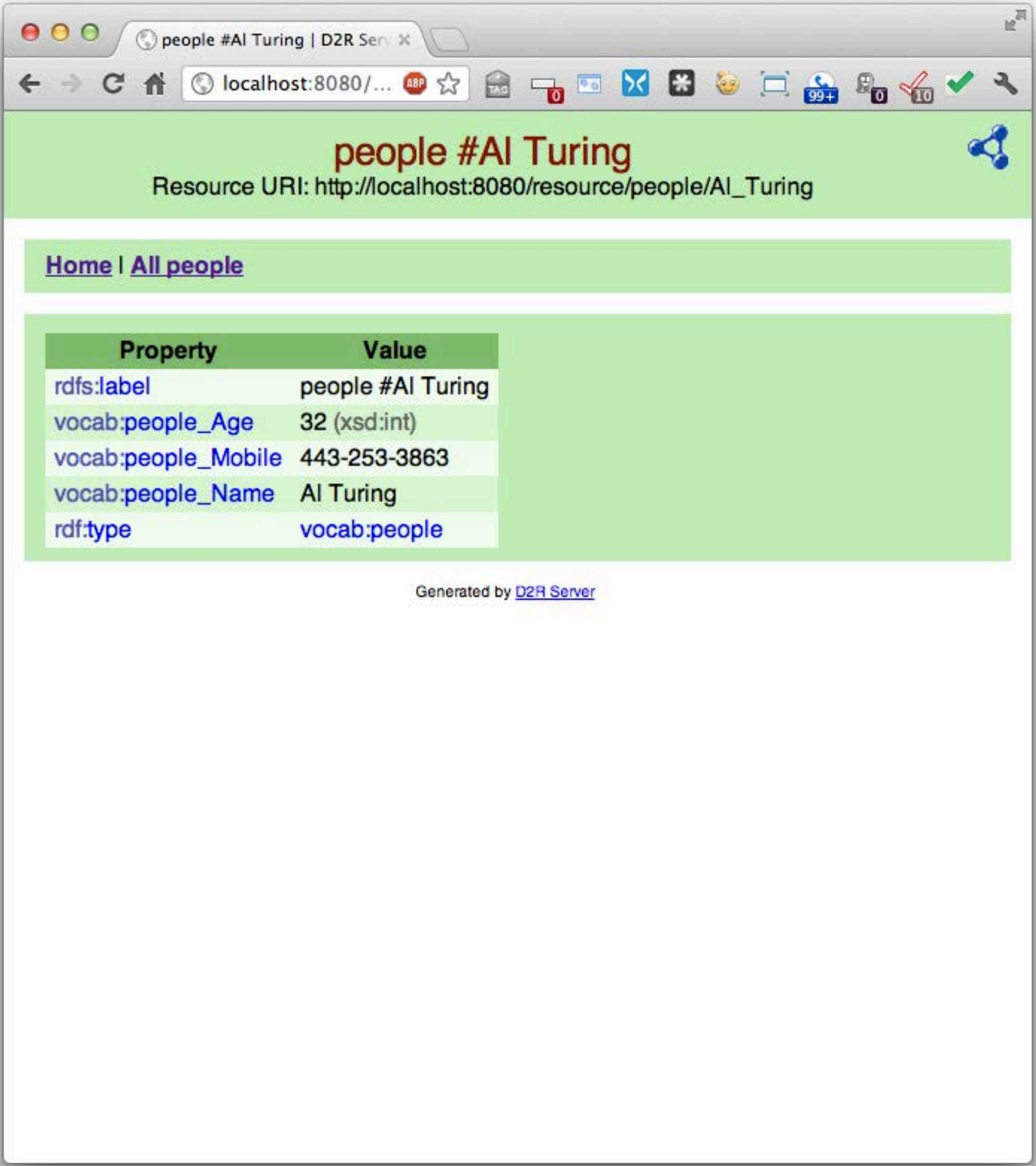
Home | people

- people #Al Turing
  http://localhost:8080/resource/people/Al_Turing
- people #Chuck Babbage
  http://localhost:8080/resource/people/Chuck_Babbage
- people #Don Knuth
  http://localhost:8080/resource/people/Don_Knuth

Generated by D2R Server

Access via D2R server

- Explore via HTML
- Via SPARQL endpoint



people #Al Turing
Resource URI: http://localhost:8080/resource/people/Al_Turing

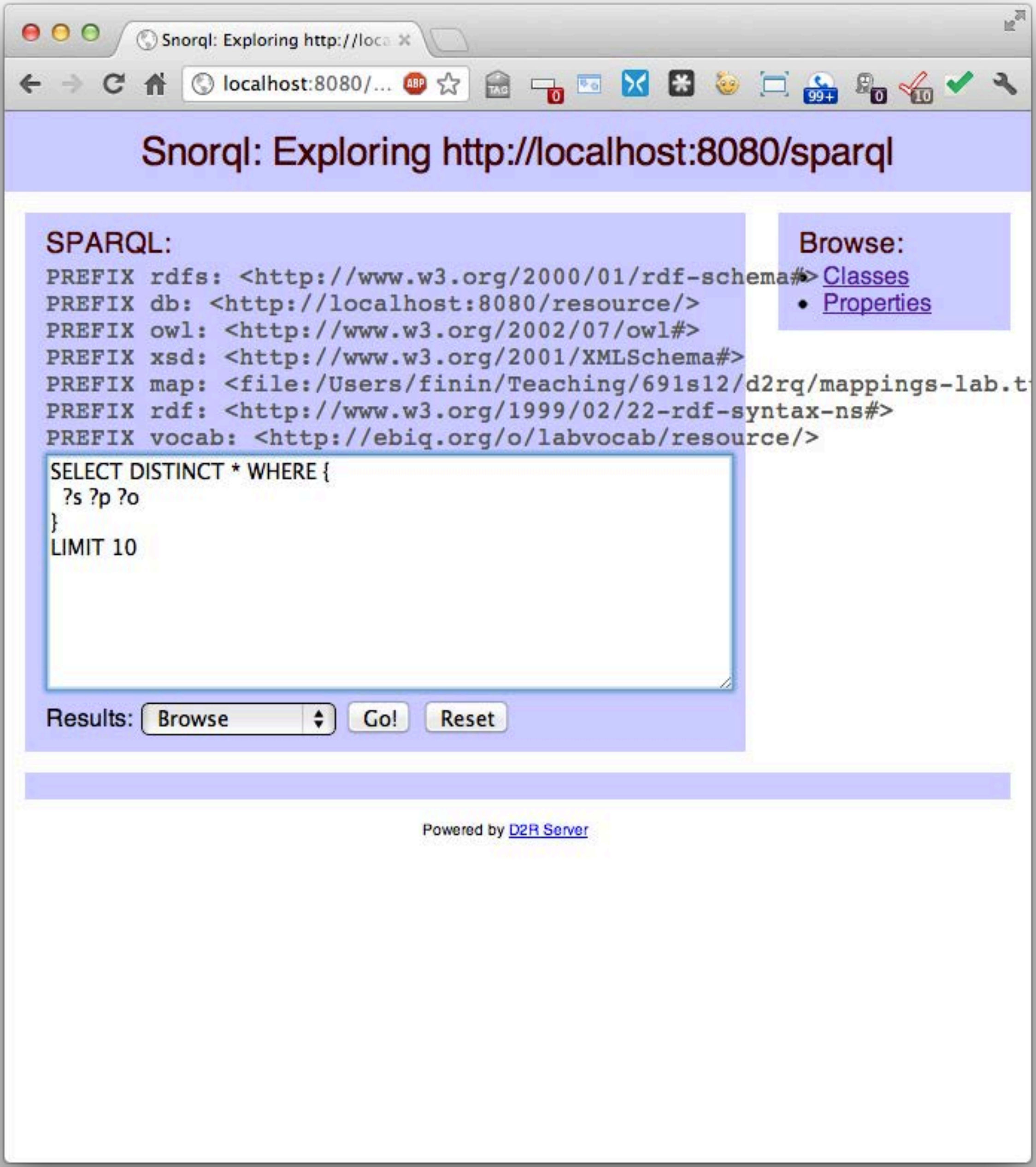
Home | All people

Property	Value
rdfs:label	people #Al Turing
vocab:people_Age	32 (xsd:int)
vocab:people_Mobile	443-253-3863
vocab:people_Name	Al Turing
rdf:type	vocab:people

Generated by D2R Server

Access via D2R server

Via SPARQL endpoint



Snorql: Exploring http://localhost:8080/sparql

SPARQL:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://localhost:8080/resource/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX map: <file:/Users/finin/Teaching/691s12/d2rq/mappings-lab.t
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX vocab: <http://ebiq.org/o/labvocab/resource/>
```

```
SELECT DISTINCT * WHERE {
  ?s ?p ?o
}
LIMIT 10
```

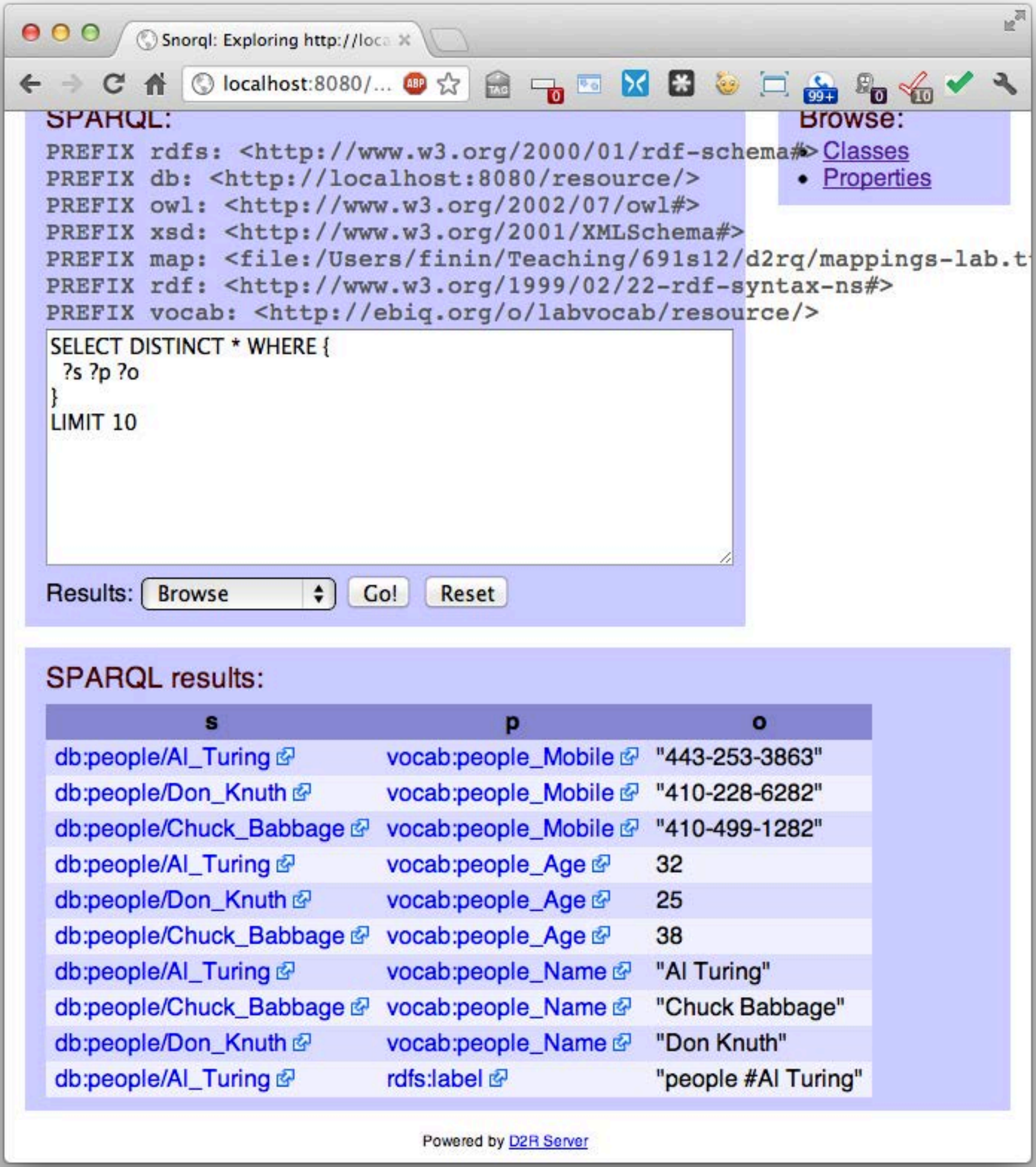
Results: Browse Go! Reset

Browse:
- Classes
- Properties

Powered by D2R Server

Access via D2R server

Via SPARQL endpoint



```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://localhost:8080/resource/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX map: <file:/Users/finin/Teaching/691s12/d2rq/mappings-lab.t
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX vocab: <http://ebiq.org/o/labvocab/resource/>
SELECT DISTINCT * WHERE {
  ?s ?p ?o
}
LIMIT 10
```

Results: Browse   Go!   Reset

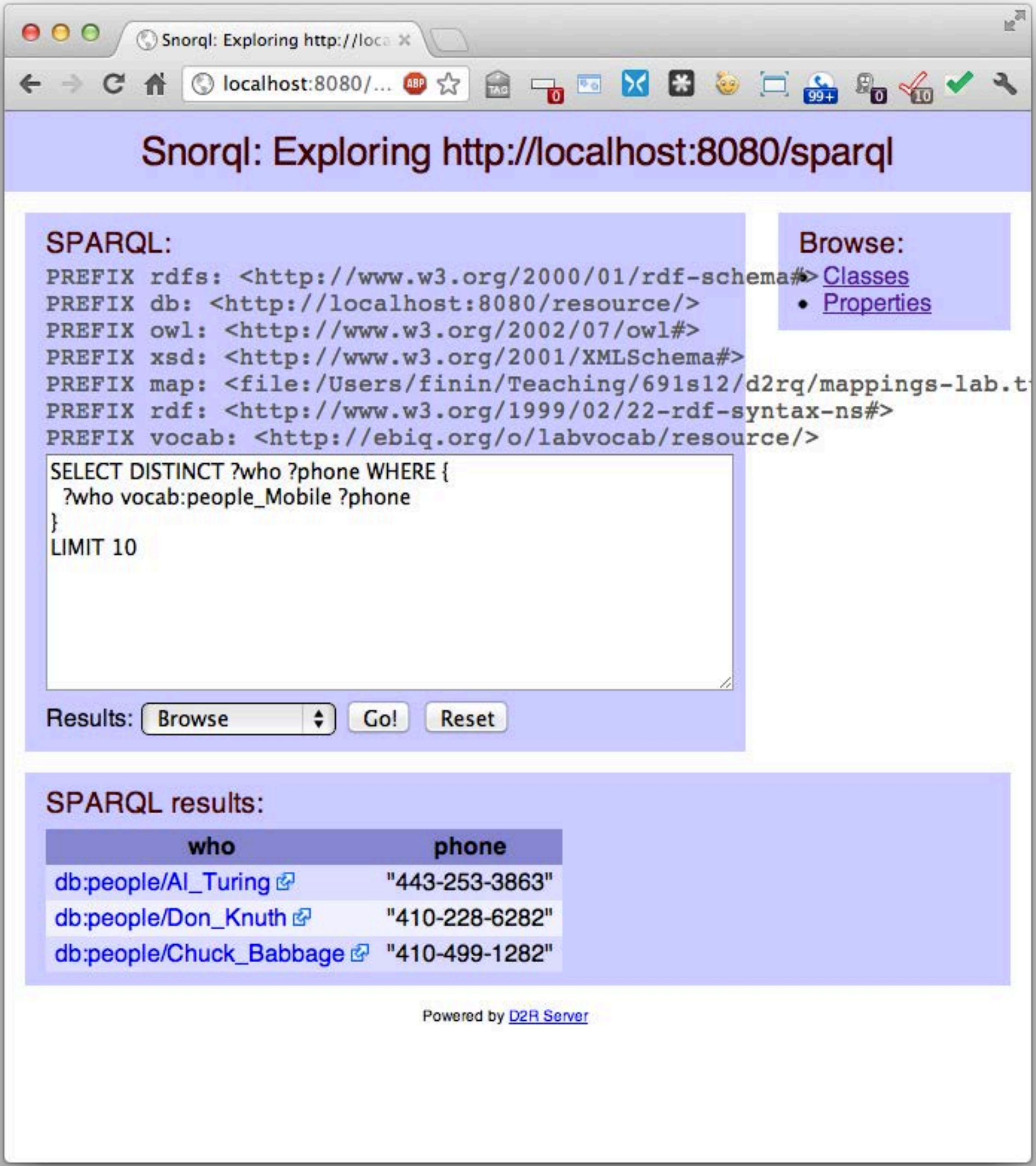
SPARQL results:

s	p	o
db:people/Al_Turing	vocab:people_Mobile	"443-253-3863"
db:people/Don_Knuth	vocab:people_Mobile	"410-228-6282"
db:people/Chuck_Babbage	vocab:people_Mobile	"410-499-1282"
db:people/Al_Turing	vocab:people_Age	32
db:people/Don_Knuth	vocab:people_Age	25
db:people/Chuck_Babbage	vocab:people_Age	38
db:people/Al_Turing	vocab:people_Name	"Al Turing"
db:people/Chuck_Babbage	vocab:people_Name	"Chuck Babbage"
db:people/Don_Knuth	vocab:people_Name	"Don Knuth"
db:people/Al_Turing	rdfs:label	"people #Al Turing"

Powered by D2R Server

Access via D2R server

Via SPARQL endpoint



Snorql: Exploring http://localhost:8080/sparql

SPARQL:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://localhost:8080/resource/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX map: <file:/Users/finin/Teaching/691s12/d2rq/mappings-lab.t
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX vocab: <http://ebiq.org/o/labvocab/resource/>
```

```
SELECT DISTINCT ?who ?phone WHERE {
  ?who vocab:people_Mobile ?phone
}
LIMIT 10
```

Results: Browse Go! Reset

Browse:
- Classes
- Properties

SPARQL results:

who	phone
db:people/Al_Turing	"443-253-3863"
db:people/Don_Knuth	"410-228-6282"
db:people/Chuck_Babbage	"410-499-1282"

Powered by D2R Server

Generating RDF dumps

Once mapping is defined, use dump-rdf for RDF dumps in various formats, e.g.:

```
% dump-rdf --w3c -o ../lab.ttl \
    -f TURTLE ../lab_map.ttl
```

Generating RDF dumps

```
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
...
@prefix vocab:   <file:///Users/finin/Sites/691f16/examples/d2rq/vocab/> .
@prefix map:     <file:///Users/finin/Sites/691f16/examples/d2rq/lab.ttl#> .
@prefix db:      <file:///Users/finin/Sites/691f16/examples/d2rq/lab.ttl> .

vocab:people_Name a rdf:Property ;
     rdfs:label "people Name" .

db:l#people/Al_Turing> a  vocab:people ;
     rdfs:label "people #Al Turing" ;
     vocab:people_Age 32 ;
     vocab:people_Mobile "443-253-3863" ;
     vocab:people_Name "Al Turing" .
…
```

Content Negotiation

- D2RQ automatically recognizes URIs for
  - Entities (e.g., an RDF object like a class or instance) http://localhost:8080/**resource**/people/Al_Turing
  - RDF representations http://localhost:8080/**data**/people/Al_Turing
  - HTML representations http://localhost:8080/**page**/people/Al_Turing
- The HTTP protocol supports *content negotiation*
- A get request can specify what kind of content it wants, e.g., HTML or RDF

Resources and 303 redirects

- Asking for raw resource make no sense – it’s just an identifier
- Client specifies in HTTP header the kind of content desired, e.g. HTML or RDF
- Server responds with an 303 redirect indicating where to go
- When client gets the 303 response, it asks for new URL

Resources and 303 redirects

```
% curl -H "Accept: text/html"  http://localhost:8081/resource/people/Al_Turing
```

303 See Other: For a description of this item, see http://localhost:8081/page/people/Al_Turing

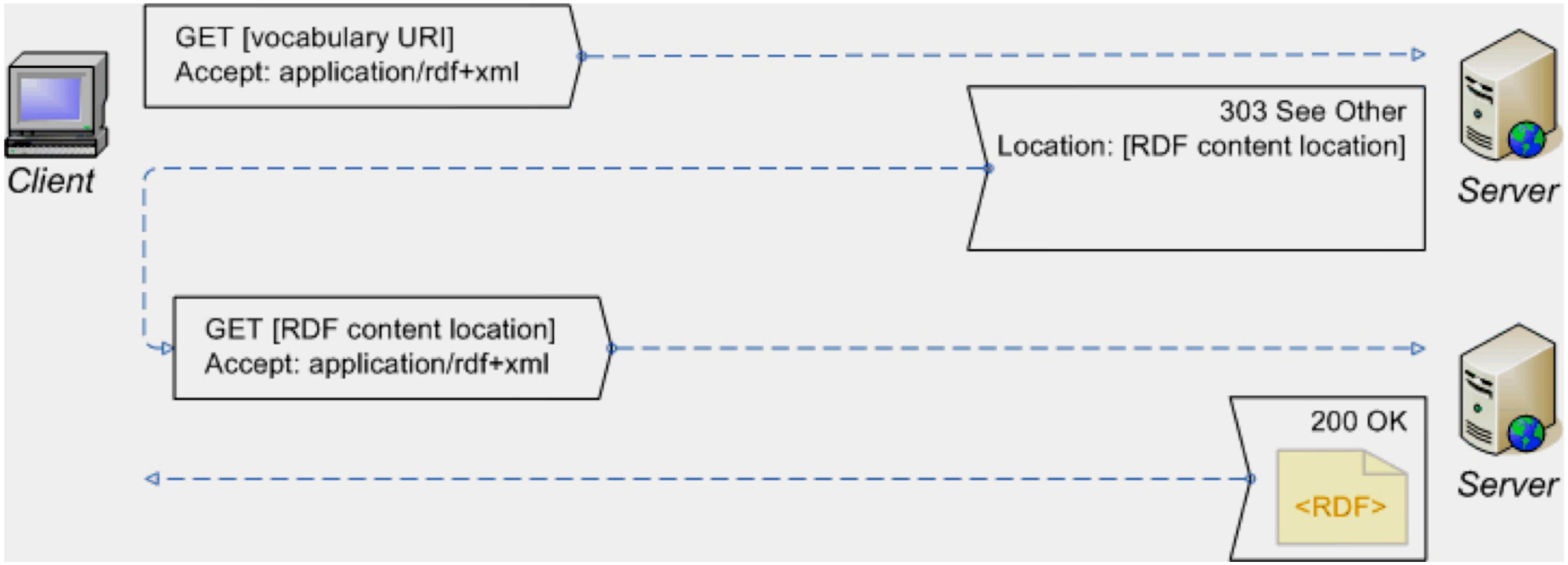
```
% curl -H "Accept: application/rdf+xml” http://localhost:8081/resource/people/Al_Turing
```

303 See Other: For a description of this item, see http://localhost:8081/data/people/Al_Turing

URIs should be de-referenceable

Linked Data best practice says that URIs should be dereferenceable;

- Doing a GET on one should always yield **useful information**


GET [vocabulary URI]
Accept: application/rdf+xml
Client
303 See Other
Location: [RDF content location]
Server
GET [RDF content location]
Accept: application/rdf+xml
200 OK
<RDF>
Server

Asking for RDF data

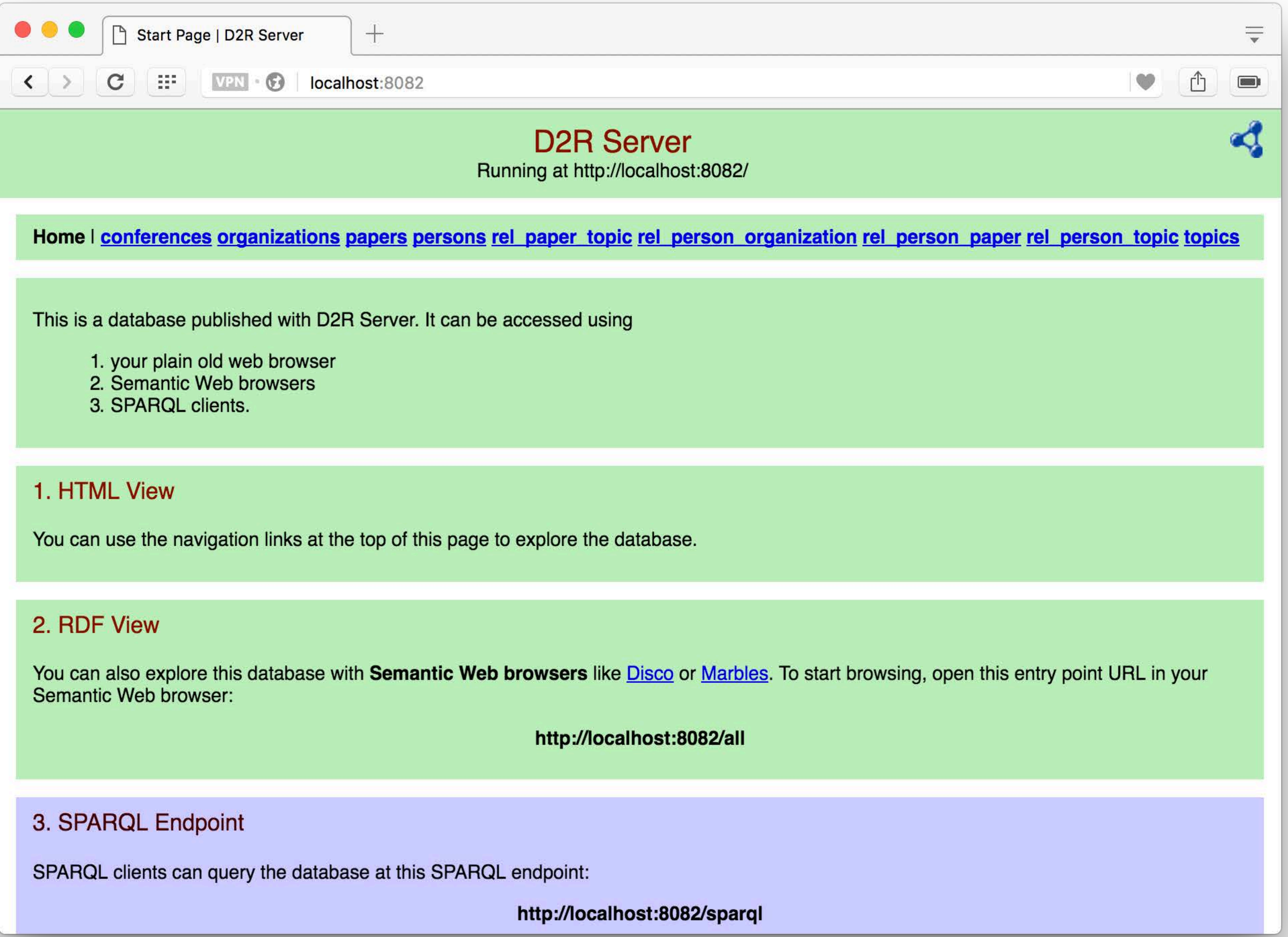
```
% curl http://localhost:8081/data/people/Al_Turing
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> . …
@prefix vocab:   <http://ebiq.org/o/labvocab/resource/> .

<http://localhost:8080/data/people/Al_Turing>
      rdfs:label "RDF Description of people #Al Turing" ;
      foaf:primaryTopic <http://localhost:8080/resource/people/Al_Turing> .

vocab:people
      rdfs:seeAlso <http://localhost:8080/sparql?query=DESCRIBE+%3Chttp%3A
%2F%2Febiq.org%2Fo%2Flabvocab%2Fresource%2Fpeople%3E> .

<http://localhost:8080/resource/people/Al_Turing>
      a       vocab:people ;
      rdfs:label "people #Al Turing" ;
      vocab:people_Age "32"^^xsd:int ;
      vocab:people_Mobile "443-253-3863" ;
      vocab:people_Name "Al Turing" .
```

Asking for HTML

```
% curl http://localhost:8081/page/people/Al_Turing
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title> people #Al Turing | D2R Server </title>
    <link rel="stylesheet" type="text/css" href="http://localhost:8080/snorql/
style.css" />
    <link rel="alternate" type="application/rdf+xml" href="http://localhost:8080/
data/people/Al_Turing?output=rdfxml" title="This page in RDF (XML)" />
    <link rel="alternate" type="text/rdf+n3" href="http://localhost:8080/data/people/
Al_Turing?output=n3" title="This page in RDF (N3)" />
  </head>
 …
```

ISWC database example

- D2RQ comes with a partial example database and mapping for information about the first ISWC conference
- To run:
  - d2r-server -port 8082 ../iswc_map.ttl
  - Visit http://localhist:8082/



D2R Server

Running at http://localhost:8082/

Home | conferences organizations papers persons rel_paper_topic rel_person_organization rel_person_paper rel_person_topic topics

This is a database published with D2R Server. It can be accessed using

1. your plain old web browser
2. Semantic Web browsers
3. SPARQL clients.

1. HTML View

You can use the navigation links at the top of this page to explore the database.

2. RDF View

You can also explore this database with **Semantic Web browsers** like Disco or Marbles. To start browsing, open this entry point URL in your Semantic Web browser:

http://localhost:8082/all

3. SPARQL Endpoint

SPARQL clients can query the database at this SPARQL endpoint:

http://localhost:8082/sparql

ISWC Database

- Information about several conferences
- It’s richer schema goes beyond a simple auto generated mapping
- This shows how to install on your com-puter and some sample queries

```
mysql> use iswc; show tables;
+--------------------------+
| Tables_in_iswc           |
+--------------------------+
| conferences              |
| organizations            |
| papers                   |
| persons                  |
| rel_paper_topic          |
| rel_person_organization  |
| rel_person_paper         |
| rel_person_topic         |
| topics                   |
+--------------------------+
9 rows in set (0.00 sec)
```