Linked (Open) Data for (Digital) Humanists

March 28, 2017 @ 10AM
http://bit.ly/dscvrdata8

About this presentation:

- Lecture: 10AM-11AM
  - Right now!
  - Mostly me talking
    - Interrupt whenever
  - Q&A at the end
  - Get a foundational understanding of LOD
    - What is it
    - Why is it important
    - Should I use it
- Workshop: 11AM-12PM
  - Totally optional
  - No software installations
  - Get hands-on experience with LOD

About this presenter:

- Hi, I’m Bryan J. Brown!
  - @bryjbrown on Twitter/GitHub
  - Repository Developer at FSU Libraries
    - Mostly working on DigiNole, FSU’s Open Source Repository
  - More DI (Digital Infrastructure) than DH
    - Let’s build platforms that enable DH work!
  - Relatively new to LOD work
    - Working on CLAW (next-gen LOD-based repository system)
    - Very high learning curve :(
    - Definitely definitely definitely the future tho (definitely!)
  - bjbrown@fsu.edu
    - Get @ me if you have questions or project ideas!

Part I:
Learning a LOD about Linked Open Data!

Chapter 1:
Vocab Lesson!

Terms to watch out for:

- Linked Data / Linked Open Data
- Semantic Web / Semantic Web Application
- Web 3.0 / Web of Data
- Triple / Triplestore
- Query
- Ontology

Terms to watch out for:

- **Linked Data / Linked Open Data**
- Semantic Web / Semantic Web Application
- Web 3.0 / Web of Data
- Triple / Triplestore
- Query
- Ontology

Terms to watch out for:

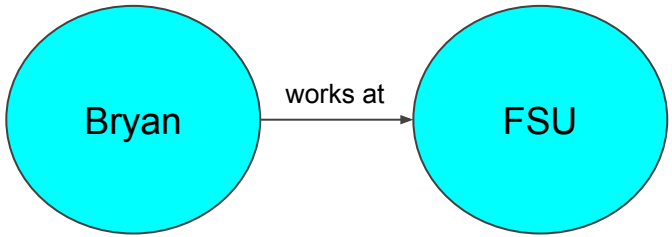
- **Linked Data / Linked Open Data**

Data that is structured by using “pointers” to connect “things” so that they all “link” to each other.

Terms to watch out for:

- **Linked Data / Linked Open Data**

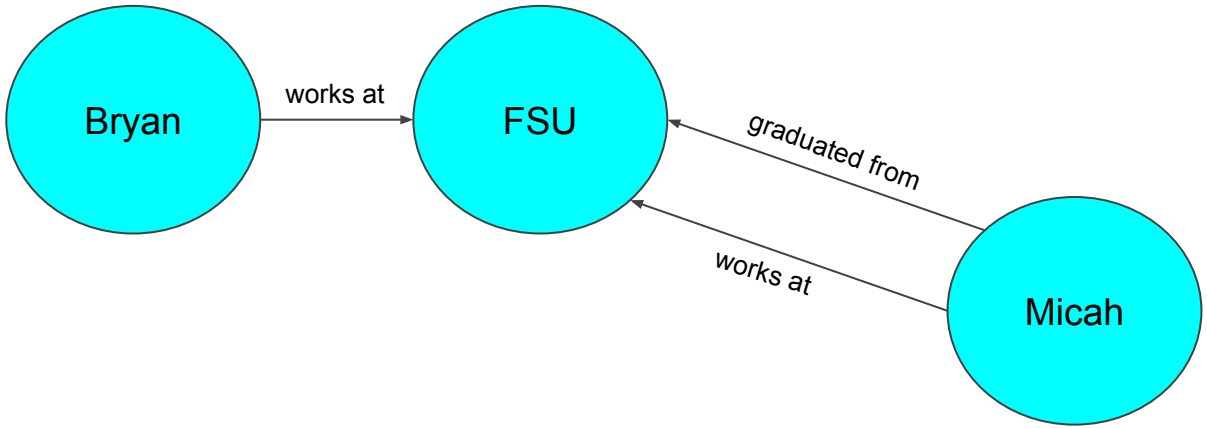
Data that is structured by using “pointers” to connect “things” so that they all “link” to each other.



Terms to watch out for:

- **Linked Data / Linked Open Data**

Data that is structured by using “pointers” to connect “things” so that they all “link” to each other.



Terms to watch out for:

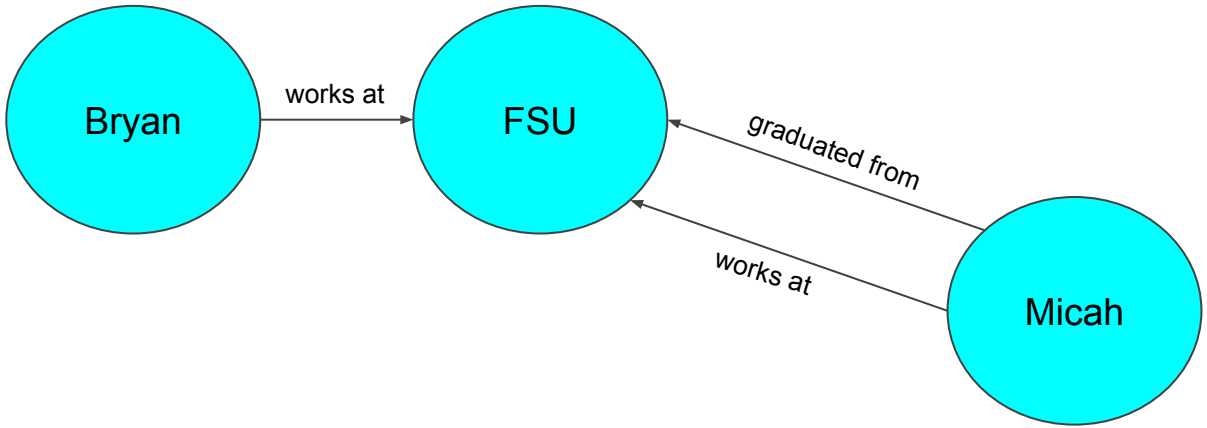
- **Linked Data / Linked Open Data**

Data that is structured by using “pointers” to connect “things” so that they all “link” to each other.

Open = freely available / not private.

Data can be linked but not open if it isn’t publicly accessible.

(#OpenData)



Terms to watch out for:

- Linked Data / Linked Open Data
- **Semantic Web / Semantic Web Application**
- Web 3.0 / Web of Data
- Triple / Triplestore
- Query
- Ontology

Terms to watch out for:

- **Semantic Web / Semantic Web Application**

Linked data is the data itself, Semantic Web is the environment in which it lives. Parts of the current web are already semantic, but most of it is not.

Opposite of “Document Web” where data is trapped in Word Docs, spreadsheets, or even HTML (can be displayed but must be scraped to make usable, and this is not reliable).

SemWeb Applications are web apps that make use of semantic web technology (some digital repositories are semweb apps).

Terms to watch out for:

- Linked Data / Linked Open Data
- Semantic Web / Semantic Web Application
- **Web 3.0 / Web of Data**
- Triple / Triplestore
- Query
- Ontology

Terms to watch out for:

- **Web 3.0 / Web of Data**

Squishy words that really just mean Semantic Web. I don’t like them because they aren’t as strictly defined.

Terms to watch out for:

- Linked Data / Linked Open Data
- Semantic Web / Semantic Web Application
- Web 3.0 / Web of Data
- **Triple / Triplestore**
- Query
- Ontology

Terms to watch out for:

- **Triple / Triplestore**

Triple = 3 part machine readable statement, the atom of linked data’s chemistry. All linked data is made of interlinked triples.

Triplestore = a database for storing triples so that you can query them.

Terms to watch out for:

- Linked Data / Linked Open Data
- Semantic Web / Semantic Web Application
- Web 3.0 / Web of Data
- Triple / Triplestore
- **Query**
- Ontology

Terms to watch out for:

- **Query**

Query = a “question” that you can ask a database.

Machine-readable data requires machine-readable questions for machine-readable answers.

Terms to watch out for:

- Linked Data / Linked Open Data
- Semantic Web / Semantic Web Application
- Web 3.0 / Web of Data
- Triple / Triplestore
- Query
- **Ontology**

Terms to watch out for:

- **Ontology**

A special specific vocabulary for expressing linked data statements. Sometimes there aren’t enough LD nouns and verbs to express exactly what you want to say, so you’ve got to build them yourself.

Example: PCDM (Portland Common Data Model), an ontology for describing digital repository stuff. Has nouns for collections & files, verbs for different types of containment.

Alphabet Soup:

- XML
- RDF / RDFS
- SPARQL
- URN / URI / URL

Alphabet Soup:

- **XML**
- RDF / RDFS
- SPARQL
- URN / URI / URL

Alphabet Soup:

- **XML**

eXtensible Markup Language. Like HTML, but you make up your own tags. Old, verbose, troublesome, but mature and popular.

I don’t like it, but w/e.

```
<SampleXML>
  <Colors>
    <Color1>White</Color1>
    <Color2>Blue</Color2>
    <Color3>Black</Color3>
    <Color4 Special="Light">Green</Color4>
    <Color5>Red</Color5>
  </Colors>
  <Fruits>
    <Fruits1>Apple</Fruits1>
    <Fruits2>Pineapple</Fruits2>
    <Fruits3>Grapes</Fruits3>
    <Fruits4>Melon</Fruits4>
  </Fruits>
</SampleXML>
```

Alphabet Soup:

- XML
- **RDF / RDFS**
- SPARQL
- URN / URI / URL

Alphabet Soup:

- **RDF / RDFS**

RDF = Resource Description Framework, defines how linked data works. Linked data is expressed through RDF.

RDFS = RDF Schema, defines how to you can create new linked data ontologies.

Alphabet Soup:

- XML
- RDF / RDFS
- **SPARQL**
- URN / URI / URL

Alphabet Soup:

- **SPARQL**

SPARQL Protocol and RDF Query Language (“sparkle”)

A query language for Linked Data. Linked data is only useful in that you can ask it really good questions, and SPARQL is how you do it.

Like SQL for triplestores.

Alphabet Soup:

- XML
- RDF / RDFS
- SPARQL
- **URN / URI / URL**

Alphabet Soup:

- **URN / URI / URL**

Uniform Resource [Name / Identifier / Locator]

A very special and specific way of “naming” something as a linked data “noun”

URL = A pointer to something on the web

URN = A unique name that doesn’t live on the web

URI = URL + URN

Alphabet Soup:

- **URN / URI / URL**

Linked data needs URIs to make sure we are all talking about the same thing. We need to be very specific.

“Bryan Brown” :(

“Bryan J. Brown, FSU Libraries” :(

“https://www.lib.fsu.edu/users/bjbrown” :)

Chapter 2:
The Humble Triple!

Making Machine-Readable Facts

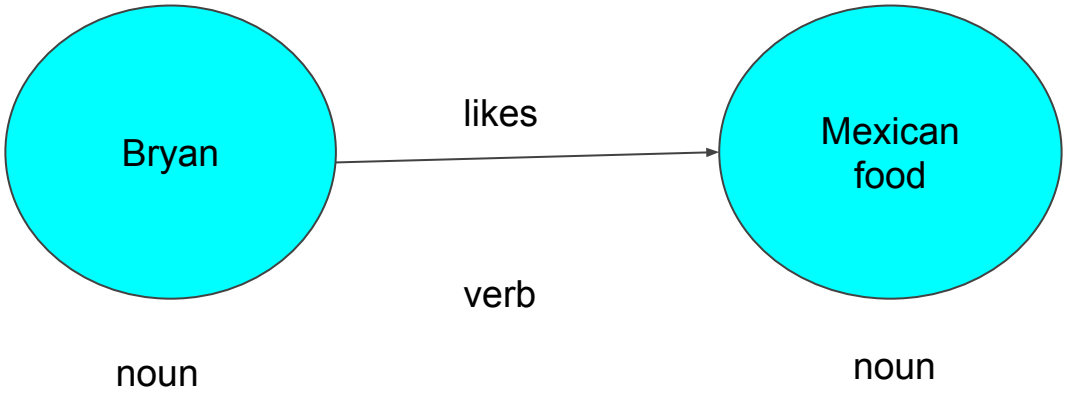
- Nouns & verbs

Nouns: “Things” on the web

Verbs: The relationship between these things

“Bryan likes mexican food.”

Making Machine-Readable Facts



Making Machine-Readable Facts

- Nouns & verbs reloaded!

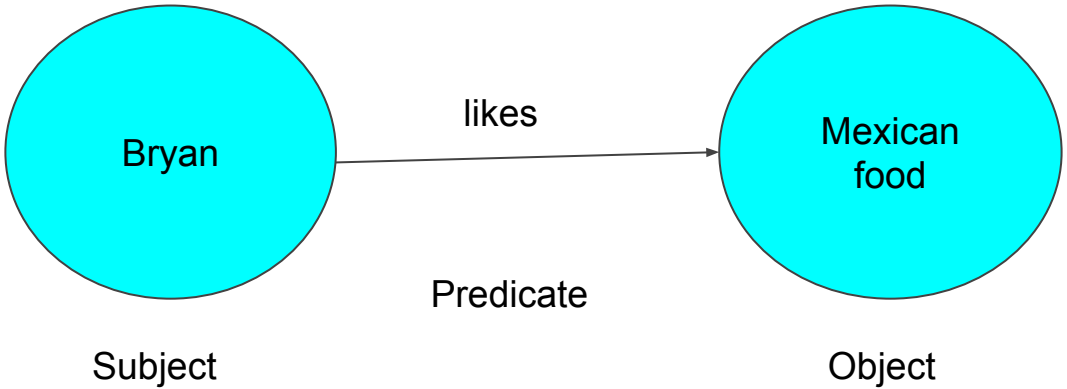
Subject: The noun the does the verbing

Predicate: The verb

Object: The noun getting verbed

These are the REAL words SemWeb folks use to talk about triples.

Making Machine-Readable Facts



Making Machine-Readable Facts

Making Machine-Readable Facts

<https://www.lib.fsu.edu/users/bjbrown>

<http://purl.org/spar/cito/likes>

<https://en.wikipedia.org/wiki/Mexican_cuisine>

Making Machine-Readable Facts

- Prefixes! Namespaces by any other name.

Really just there to shorten these long URIs.

<https://en.wikipedia.org/wiki/Mexican_cuisine>

~ BECOMES ~

wiki: <https://en.wikipedia.org/wiki/>

wiki:Mexican_cuisine

Making Machine-Readable Facts

```
fsulib: <https://www.lib.fsu.edu/users/>
cito: <http://purl.org/spar/cito/>
wiki: <https://en.wikipedia.org/wiki/>

<fsulib:bjbrown> <cito:likes> <wiki:Mexican_cuisine>
```

Making Machine-Readable Facts

```
fsulib: <https://www.lib.fsu.edu/users/>
cito: <http://purl.org/spar/cito/>
wiki: <https://en.wikipedia.org/wiki/>

<fsulib:bjbrown> <cito:likes> <wiki:Mexican_cuisine>
```

Way simpler, especially when you’re about to define a lot of similar triples.

Making Machine-Readable Facts

```
fsulib: <https://www.lib.fsu.edu/users/>
cito: <http://purl.org/spar/cito/>
wiki: <https://en.wikipedia.org/wiki/>

<fsulib:bjbrown> <cito:likes> <wiki:Mexican_cuisine>
<fsulib:bjbrown> <cito:likes> <wiki:Halloween>
```

Making Machine-Readable Facts

```
fsulib: <https://www.lib.fsu.edu/users/>
cito: <http://purl.org/spar/cito/>
wiki: <https://en.wikipedia.org/wiki/>

<fsulib:bjbrown> <cito:likes> <wiki:Mexican_cuisine>
<fsulib:bjbrown> <cito:likes> <wiki:Halloween>
<fsulib:bjbrown> <cito:likes> <wiki:Science_fiction>
```

Making Machine-Readable Facts

```
fsulib: <https://www.lib.fsu.edu/users/>
cito: <http://purl.org/spar/cito/>
wiki: <https://en.wikipedia.org/wiki/>

<fsulib:bjbrown> <cito:likes> <wiki:Mexican_cuisine>
<fsulib:bjbrown> <cito:likes> <wiki:Halloween>
<fsulib:bjbrown> <cito:likes> <wiki:Science_fiction>
<fsulib:mvandegrift> <cito:likes> <wiki:Mexican_cuisine>
```

Making Machine-Readable Facts

```
fsulib: <https://
cito: <http://pu
wiki: <https://en

<fsulib:bjbrown> <
<fsulib:bjbrown> <ci
<fsulib:bjbrown> <cit
<fsulib:mvandegrift> <c
<fsulib:bjbrown> <cito:li
```

Making Machine-Readable Facts

“Nouns” and “Verbs” aren’t “real” vocab words used by SemWeb folks, but it’s an easy way to explain their role.

The subject of one triple can be the object of another! A predicate can only be a subject or object in VERY SPECIAL CASES (ontology definitions).

Chapter 3:
Structured Data = Structure Questions!

Let’s get SPARQLing

———

SPARQL queries can be understood as expressing a pattern that a triple COULD fit into, and the query returns data that fits the pattern.

Define the thing you what returned (SELECT) and the pattern that this thing could fit (WHERE).

Additionally, you may add certain “modifiers” to filter out the returned results.

Let’s get SPARQLing

- **Who likes mexican cuisine?**

```
PREFIX cito: <http://purl.org/spar/cito/>
PREFIX wiki: <https://en.wikipedia.org/wiki/>
SELECT ?person
WHERE {
  ?person cito:likes wiki:Mexican_cuisine
}
```

Let’s get SPARQLing

```
---
- Result:
```

person
https://www.lib.fsu.edu/users/bjbrown
https://www.lib.fsu.edu/users/mvandegrift

Let’s get SPARQLing

- **Who likes mexican cuisine?**

```
PREFIX cito: <http://purl.org/spar/cito/>
PREFIX wiki: <https://en.wikipedia.org/wiki/>
SELECT ?person
WHERE {
  ?person cito:likes wiki:Mexican_cuisine
} LIMIT 1
```

Let’s get SPARQLing

```
---
- Result:
```

person
https://www.lib.fsu.edu/users/bjbrown

Let’s get SPARQLing

- **Who likes mexican cuisine?**

```
PREFIX cito: <http://purl.org/spar/cito/>
PREFIX wiki: <https://en.wikipedia.org/wiki/>
SELECT ?person
WHERE {
  ?person cito:likes wiki:Mexican_cuisine
}
ORDER BY ?person
```

Let’s get SPARQLing

```
---
- Result:
```

person
https://www.lib.fsu.edu/users/bjbrown
https://www.lib.fsu.edu/users/mvandegrift

Let’s get SPARQLing

- **Who likes mexican cuisine?**

```
PREFIX cito: <http://purl.org/spar/cito/>
PREFIX wiki: <https://en.wikipedia.org/wiki/>
SELECT ?person
WHERE {
  ?person cito:likes wiki:Mexican_cuisine
}
ORDER BY DESC(?person)
```

Let’s get SPARQLing

```
---
- Result:
```

person
https://www.lib.fsu.edu/users/mvandegrift
https://www.lib.fsu.edu/users/bjbrown

Let’s get SPARQLing

- **Who likes mexican cuisine?**

```
PREFIX cito: <http://purl.org/spar/cito/>
PREFIX wiki: <https://en.wikipedia.org/wiki/>
SELECT ?person
WHERE {
  ?person cito:likes wiki:Mexican_cuisine
}
ORDER BY DESC(?person)
LIMIT 1
```

Let’s get SPARQLing

```
---
- Result:
```

person
https://www.lib.fsu.edu/users/mvandegrift

Let’s get SPARQLing

- **What does Bryan like?**

```
PREFIX fsulib: <https://www.lib.fsu.edu/users/>
PREFIX cito: <http://purl.org/spar/cito/>
SELECT ?thing
WHERE {
  fsulib:bjbrown cito:likes ?thing
}
```

Let’s get SPARQLing

- **Result:**

thing
https://en.wikipedia.org/wiki/Halloween
https://en.wikipedia.org/wiki/Mexican_cuisine
https://en.wikipedia.org/wiki/Science_fiction
https://www.lib.fsu.edu/users/mvandegrift

Let’s get SPARQLing

- **WHO does Bryan like?**

WHO implies that the object of Bryan’s “like” is a person. We’ll need to add a bit of data to our triplestore to let it know Micah is a person.

Let’s get SPARQLing

- **WHO does Bryan like?**

WHO implies that the object of Bryan’s “like” is a person. We’ll need to add a bit of data to our triplestore to let it know Micah is a person.

```
fsulib: <https://www.lib.fsu.edu/users/>
rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
foaf: <http://xmlns.com/foaf/0.1/>

<fsulib:mvandegrift> <rdf:type> <foaf:Person>
```

Let’s get SPARQLing

- **WHO does Bryan like?**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fsulib: <https://www.lib.fsu.edu/users/>
PREFIX cito: <http://purl.org/spar/cito/>
SELECT ?thing
WHERE {
  fsulib:bjbrown cito:likes ?thing .
  ?thing rdf:type foaf:Person
}
# The dot after the first triple pattern means the pattern
continues
```

Let’s get SPARQLing

```
---
- Result:
```

thing
https://www.lib.fsu.edu/users/mvandegrift

Let's get SPARQLing

- **WHO does Bryan like THAT ALSO LIKES MEXICAN CUISINE?**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fsulib: <https://www.lib.fsu.edu/users/>
PREFIX cito: <http://purl.org/spar/cito/>
PREFIX wiki: <https://en.wikipedia.org/wiki/>
SELECT ?thing
WHERE {
  fsulib:bjbrown cito:likes ?thing .
  ?thing rdf:type foaf:Person .
  ?thing cito:likes wiki:Mexican_cuisine
}
```

Let’s get SPARQLing

```
---
- Result:
```

thing
https://www.lib.fsu.edu/users/mvandegrift

Chapter 4:
All Together Now!

So what?

- SPARQL queries are LOD’s killer app
  - Does for information what SQL does for data
    - But doesn’t require knowledge of table and column names
    - And doesn’t require access to the database
    - And you can query every database at once
      - With the same query

So what?

- ~~SPARQL queries~~ *Triplestores* are LOD's killer app
  - SPARQL is nice, but the way triplestores store the data and provide access to it is the real magic.
    - RDF triples are how you tell the triplestore what you know
    - SPARQL is how you ask the triplestore what it knows
  - Triplestores can talk to each other as well to mirror data and execute "federated" queries on other triplestores, too
    - Not just asking the triplestore what it knows, but asking THE INTERNET what it knows
      - Words can't express how awesome this is to me

So what?

- SPARQL queries can only return known data
  - Most of the world’s truth has not been expressed as a triple and ingested into a triplestore yet, so you can’t SPARQL it
    - Maybe it has been, but it was done badly (likely)

So what?

- SPARQL queries can only return known data
  - Most of the world's truth has not been expressed as a triple and ingested into a triplestore yet, so you can't SPARQL it
    - Maybe it has been, but it was done badly (likely)

ALSO WHAT DOES ANY OF THIS HAVE TO DO WITH DIGITAL HUMANITIES?

Oh, the humanities.

———

- Semantic Web is the future of digital scholarship
  - Getting data via SPARQL will become an important tool in digital humanist's arsenal. Yet another methodology?
- Semantic Web's utility depends on availability of data
  - More Linked Open Data means more DH projects that can be done with Linked Open Data
  - Publishing your domain knowledge as LOD is a totally worthwhile endeavor
    - Also sounds impressive
      - Do it
        - srlsy


I WANT YOU
TO MAKE MORE LINKED DATA

The End!
Questions?


INTERMISSION

Part II:
Learning How to Ask Good Questions!

Chapter 5:
Say Hello to DBpedia!

DBpedia.org

- https://dbpedia.org/
- “a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web”
- A Semantic Web application
- All resources (nouns + verbs) have human-readable pages that show their linkage
- A great way to explore LOD in a visual way

DBpedia.org

- Also has a SPARQL endpoint (place to execute SPARQL queries): https://dbpedia.org/sparql
- Can be used to play with LOD by coming up with fun SPARQL queries against LOTS of real-world data.
- Also a good teacher for common mistakes in SPARQL queries
- Common gotchas:
  - Having /page/ in the URI
    - DBpedia redirects URIs (/resource/) to URLs (/page/)
    - Copy the link directly from the page when possible
  - Missing <> around URI (needs to look like <https://this.url>)
  - Keep “Results Format” on “HTML”

DBpedia.org

Smoke test: Run this query

```
SELECT ?p ?o
WHERE {
  <http://dbpedia.org/resource/Florida_State_University_School> ?p ?o
}
```

“Get every predicate and object
that has FSU as a subject.”

What do you see?

Chapter 6:
We Can’t Stop Here, This is Bro-Country!

Come at me, Bro-country

———

You have a friend who says they like “Bro-country”

- What does that even mean
  - Lets find out?

1. Find the noun!
   a. http://dbpedia.org/resource/Bro-country
2. Ask DBpedia what it knows

```
SELECT ?p ?o
WHERE {
  <http://dbpedia.org/resource/Bro-country> ?p ?o
}
```

Come at me, Bro-country

———

```
# What IS Bro-country?
SELECT ?o
WHERE {
  <http://dbpedia.org/resource/Bro-country> <http://dbpedia.org/ontology/abstract> ?o
}

# What are the stylistic origins of Bro-country?
SELECT ?o
WHERE {
  <http://dbpedia.org/resource/Bro-country> <http://dbpedia.org/property/stylisticOrigins> ?o
}

# What does Bro-country look like in practice?
SELECT ?o
WHERE {
  <http://dbpedia.org/resource/Bro-country> <http://xmlns.com/foaf/0.1/depiction> ?o
}
```

Come at me, Bro-country

Chapter 7:
Linked Open Dada!

Let’s find some artists

```
# Note: “a” as a predicate is short for rdf:type
SELECT ?artist
WHERE {
  ?artist a <http://dbpedia.org/ontology/Artist>
}
```

Let’s find some artists

———

```
# Artists who were Cubists
SELECT ?artist
WHERE {
  ?artist a <http://dbpedia.org/ontology/Artist> .
  ?artist <http://dbpedia.org/ontology/movement> <http://dbpedia.org/resource/Cubism>
}

# Artists who were Cubists AND Dadaists
SELECT ?artist
WHERE {
  ?artist a <http://dbpedia.org/ontology/Artist> .
  ?artist <http://dbpedia.org/ontology/movement> <http://dbpedia.org/resource/Cubism> .
  ?artist <http://dbpedia.org/ontology/movement> <http://dbpedia.org/resource/Dada>
}
```

Let’s find some artists

———

```
# Artists who were Cubists AND Dadaists AND Spaniards
SELECT ?artist
WHERE {
  ?artist a <http://dbpedia.org/ontology/Artist> .
  ?artist <http://dbpedia.org/ontology/movement> <http://dbpedia.org/resource/Cubism> .
  ?artist <http://dbpedia.org/ontology/movement> <http://dbpedia.org/resource/Dada> .
  ?artist <http://dbpedia.org/ontology/nationality> <http://dbpedia.org/resource/Spaniards>
}
```


Hello,
Dali!

Chapter 8:
A Postmodern Feminist by Any Other Name!

Mo PoMo, fo sho

———

```
# Get all postmodern feminists
SELECT ?feminist
WHERE {
  ?feminist <http://purl.org/dc/terms/subject> <http://dbpedia.org/resource/Category:Postmodern_feminists> .
}

# Get all postmodern feminists who wrote under pseudonyms
SELECT ?feminist
WHERE {
  ?feminist <http://purl.org/dc/terms/subject> <http://dbpedia.org/resource/Category:Postmodern_feminists> .
  ?feminist <http://purl.org/dc/terms/subject> <http://dbpedia.org/resource/Category:Pseudonymous_writers>
}

# Get the pseudonym and birth name of all postmodern feminists who wrote under pseudonyms
SELECT ?pseudonym ?birthname
WHERE {
  ?feminist <http://purl.org/dc/terms/subject> <http://dbpedia.org/resource/Category:Postmodern_feminists> .
  ?feminist <http://purl.org/dc/terms/subject> <http://dbpedia.org/resource/Category:Pseudonymous_writers> .
  ?feminist <http://dbpedia.org/ontology/birthName> ?birthname .
  ?feminist <http://dbpedia.org/property/name> ?pseudonym
}
```

Ring any bells? <http://dbpedia.org/page/Bell_hooks>

In closing...

———

- This was all just to sell you on the power of LOD, not to be an exhaustive reference (though you may feel exhausted…)
- There's a lot more to LOD than just DBpedia
- There's a lot more to SPARQL than what we tried

If you think you have an idea that could benefit from querying (or producing!) linked open data, talk to us! We want to help!

https://www.lib.fsu.edu/drs/digital-humanities

The End! (again.)
Questions?