Semantic Web Working Group

Oct. 4, 2016

Review and continue with sections near 2.3

Steve Baskauf

Review and changes since last time

Table containing metadata on instances of the site class

subject resource IRI = domain root + local name

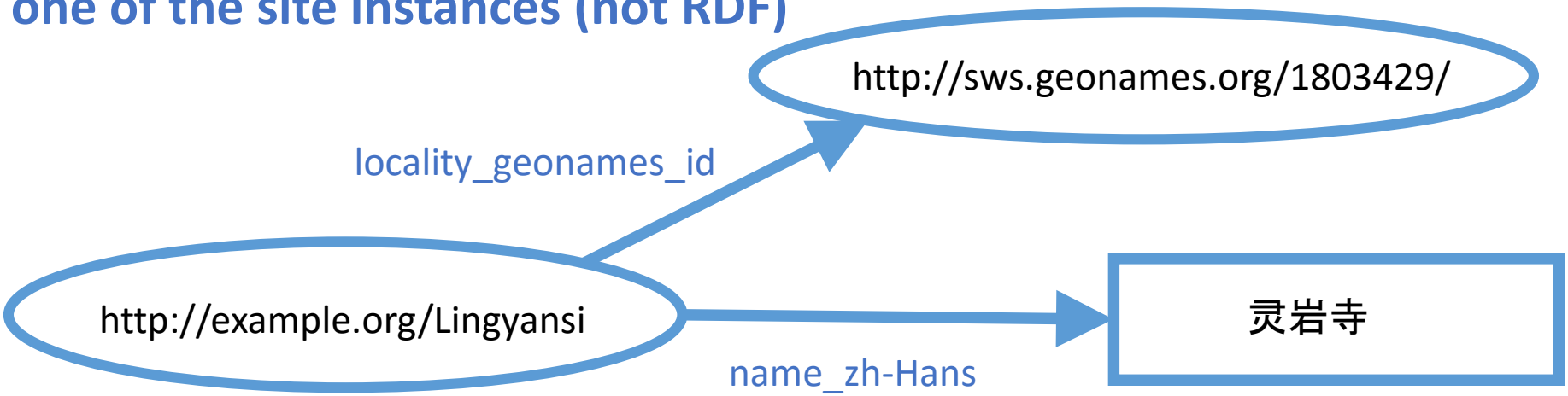
columns = properties (predicates)

iri_local_name	name_zh-Hans	locality_geonames_id
Lingyansi	灵岩寺	http://sws.geonames.org/1803429/
Longmensi	龙门寺	
Xilimen	西李门二仙庙	

rows = instances of class

cells = values (objects)

Graph representation of metadata about one of the site instances (not RDF)



Not all of the values in a row of a table are really “about” the subject of the row.

Some columns are about resources that have a 1:1 relationship to the subject resource.

Example of related classes of resources: site and time period of construction at the site.

classes.csv table for classes having a one:one relationship to the root class

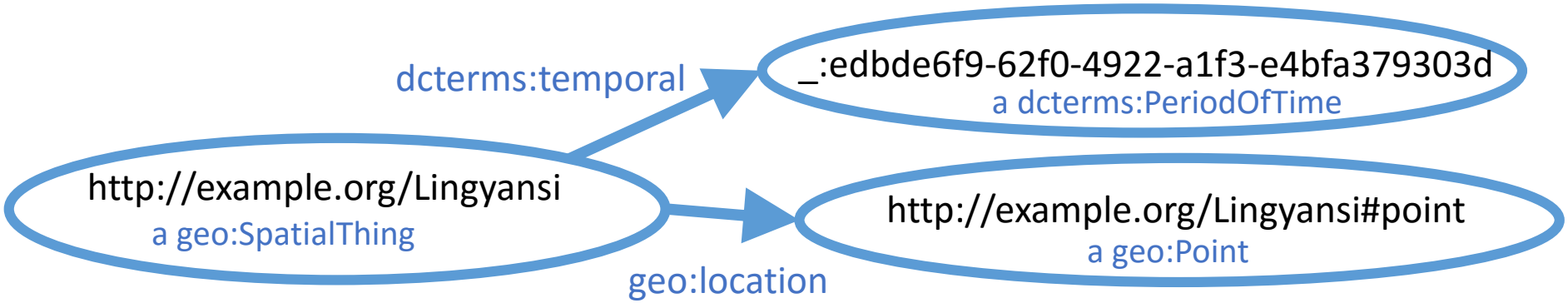
id	class
	geo:SpatialThing
_:1	dcterms:PeriodOfTime
point	geo:Point

The id column indicates modifications to the root IRI identifier for identifying instances of related resources having a 1:1 relationship whose metadata are represented in the table.

- no value means the root IRI is unmodified
- values beginning with "_:" indicate that the resource is a blank node
- other values indicate text to be appended to the root IRI as a fragment identifier

For example:

`http://example.org/Lingyansi` = the IRI for the *geo:SpatialThing* instance
`http://example.org/Lingyansi#point` = the IRI for the *geo:Point* instance
`_:edbde6f9-62f0-4922-a1f3-e4bfa379303d` = a temporary blank node identifier for the *dcterms:PeriodOfTime* instance



Creating the mappings from the tabled metadata to an RDF graph

Important properties that almost everybody uses:

rdf:type the class that the thing is an instance of
rdfs:label what people call the thing (for creative works also dcterms:title)
rdfs:comment any kind of free text about the thing
rdfs:seeAlso a generic link to any related resource, doesn't have to be machine-readable

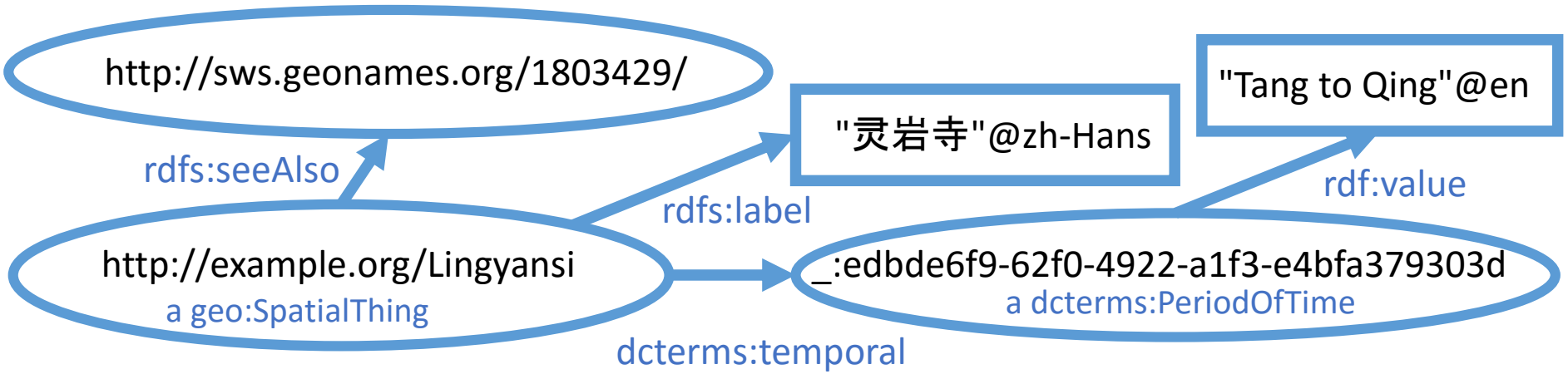
Many terms from the Dublin Core namespaces (dc: dcterms: dcmitype:) are commonly used. This is particularly true for metadata about creative works, including the RDF document itself.

metadata.csv table for the site class

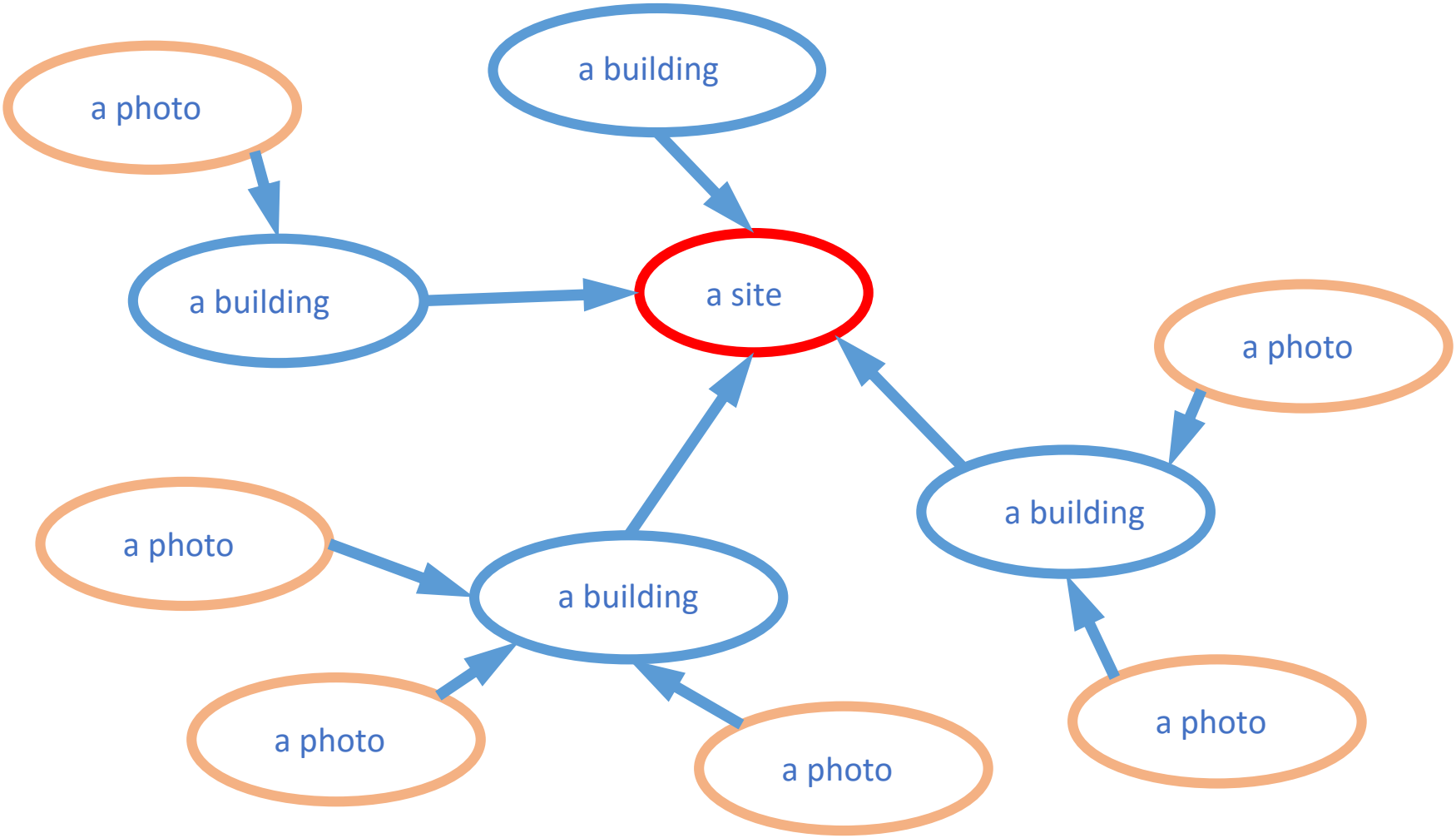
iri_local_name	name_zh-Hans	locality_geonames_id	site_date_verbatim_en
Lingyansi	灵岩寺	http://sws.geonames.org/1803429/	Tang to Qing
Longmensi	龙门寺		Song to Qing
Xilimen	西李门二仙庙		Song to Yuan

metadata-column-mappings.csv table

header	predicate	type	value	attribute	class
name_zh-Hans	rdfs:label	language		zh-Hans	geo:SpatialThing
locality_geonames_id	rdfs:seeAlso	iri			geo:SpatialThing
$link	dcterms:temporal	iri	_:1		geo:SpatialThing
site_date_verbatim_en	rdf:value	language		en	dcterms:PeriodOfTime

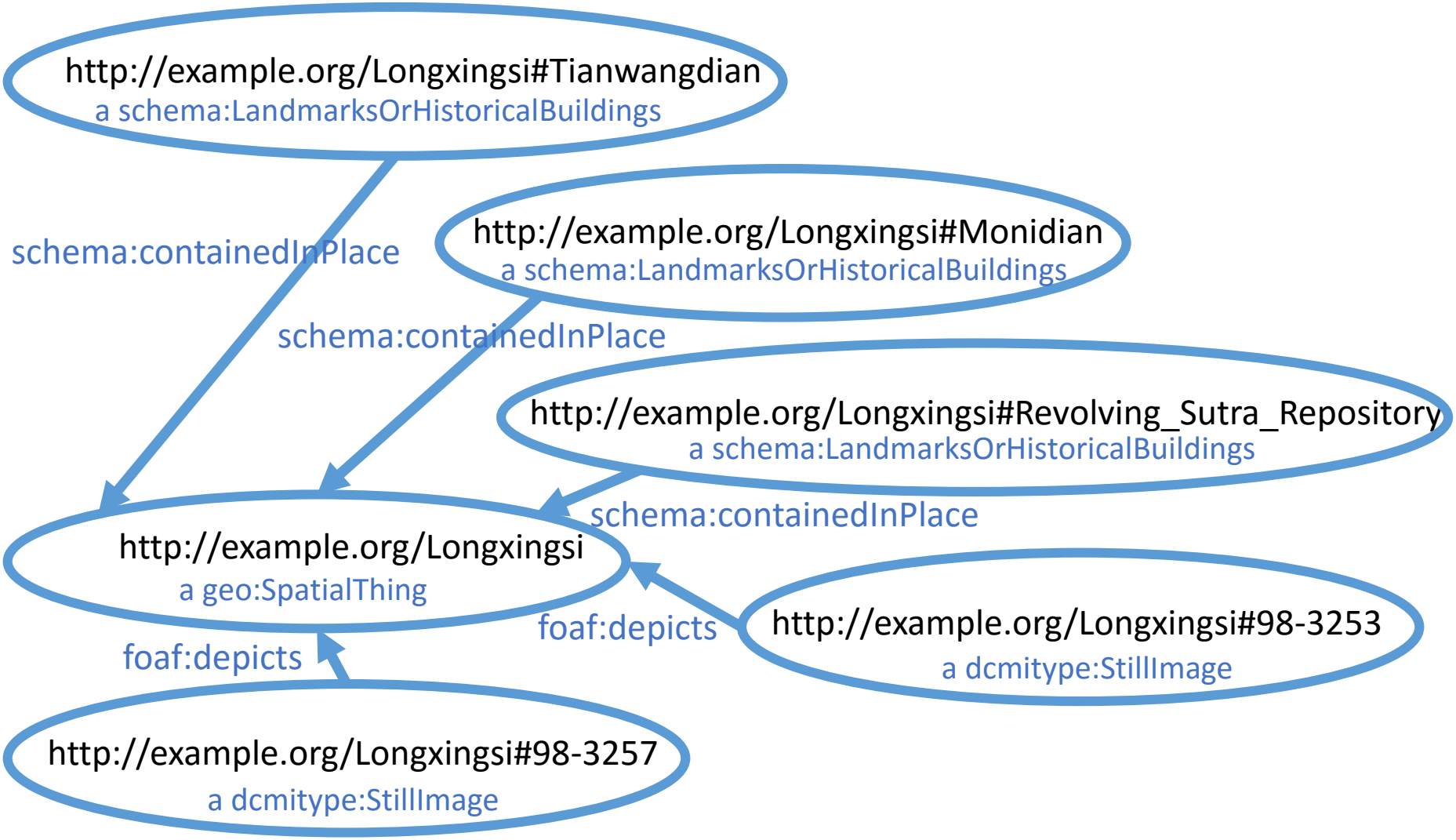


The model we want is too complicated to be represented by 1:1 relationships.
It's going to have many:1 or many:many relationships.



linked-classes.csv table for classes having a many:one relationship with the root class

link_column	link_property	suffix1	link_characters	suffix2	filename
site_name_zh-Latn-pinyin	schema:containedInPlace	building_local_name			buildings
site_id	foaf:depicts	foto_year	-	foto_num	photos

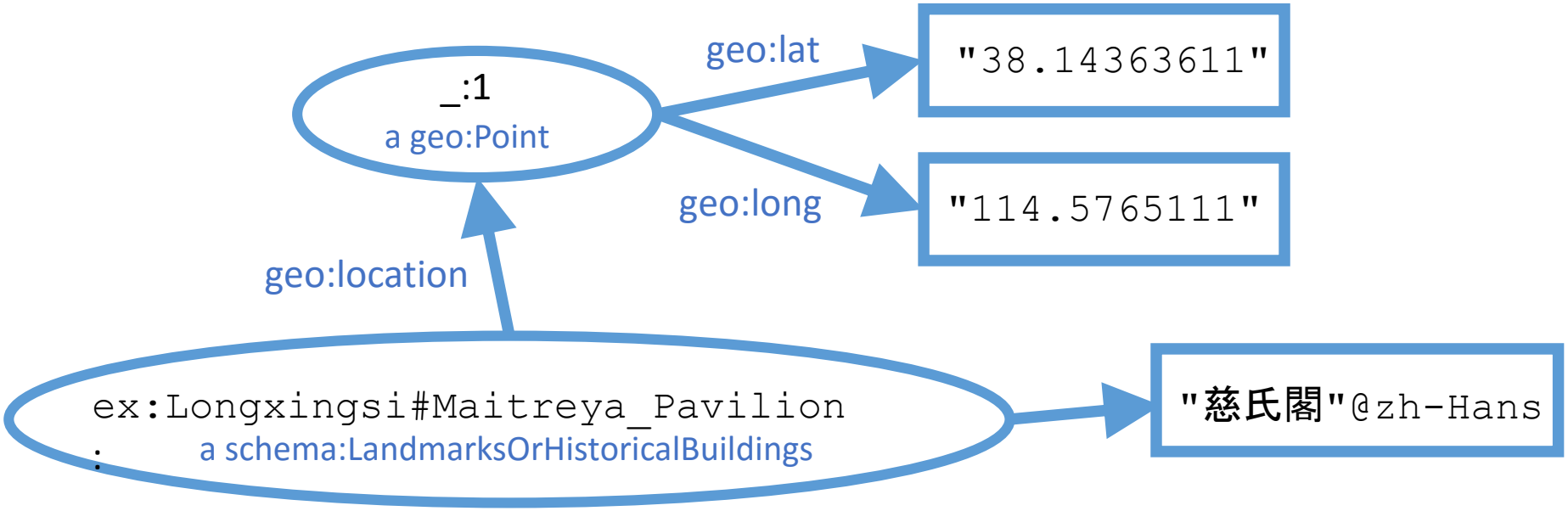


Since last time…

- Update your fork of the semantic-web repo because there are many changes to the files and scripts.
- Open test-serialize.xq in BaseX to try the new scripts.
- Open the metadata.csv file to find local names to try. It and buildings.csv have been cleaned up.
- The mapping files are at metadata-column-mappings.csv and buildings-column-mappings.csv
- See the instructions on the readme.md page for the tang-song directory

What are the right classes for the sites and buildings? How should we link them elsewhere?

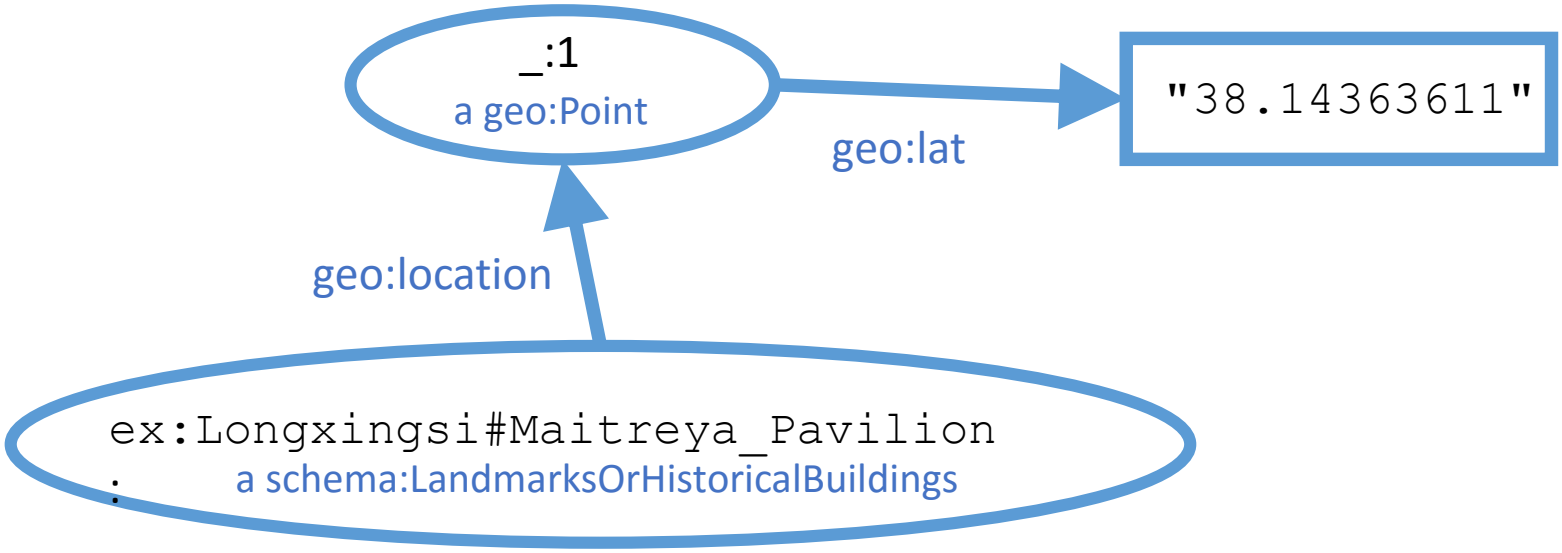
iri	name_zh-Hans	decimal_latitude	decimal_longitude
ex:Longxingsi#Maitreya_Pavilion	慈氏閣	38.14363611	114.5765111




```
ex:Longxingsi#Maitreya_Pavilion
    geo:location _:1;
    a schema:LandmarksOrHistoricalBuildings.

_:1 geo:lat "38.14363611";
    geo:long "114.5765111";
    a geo:Point.
```

Ranges and domains from
https://www.w3.org/2003/01/geo/wgs84_pos



Since the range of geo:location is geo:SpatialThing, if

```
ex:Longxingsi#Maitreya_Pavilion
    geo:location _:1.
```

then

```
_:1 a geo:SpatialThing.
```

Since the domain of geo:lat is geo:SpatialThing, if

```
_:1 geo:lat "38.14363611".
```

then

```
_:1 a geo:SpatialThing.
```

Subclass relationship from
`https://www.w3.org/2003/01/geo/wgs84_pos`

Since

```
geo:Point rdfs:subClassOf geo:SpatialThing.
```

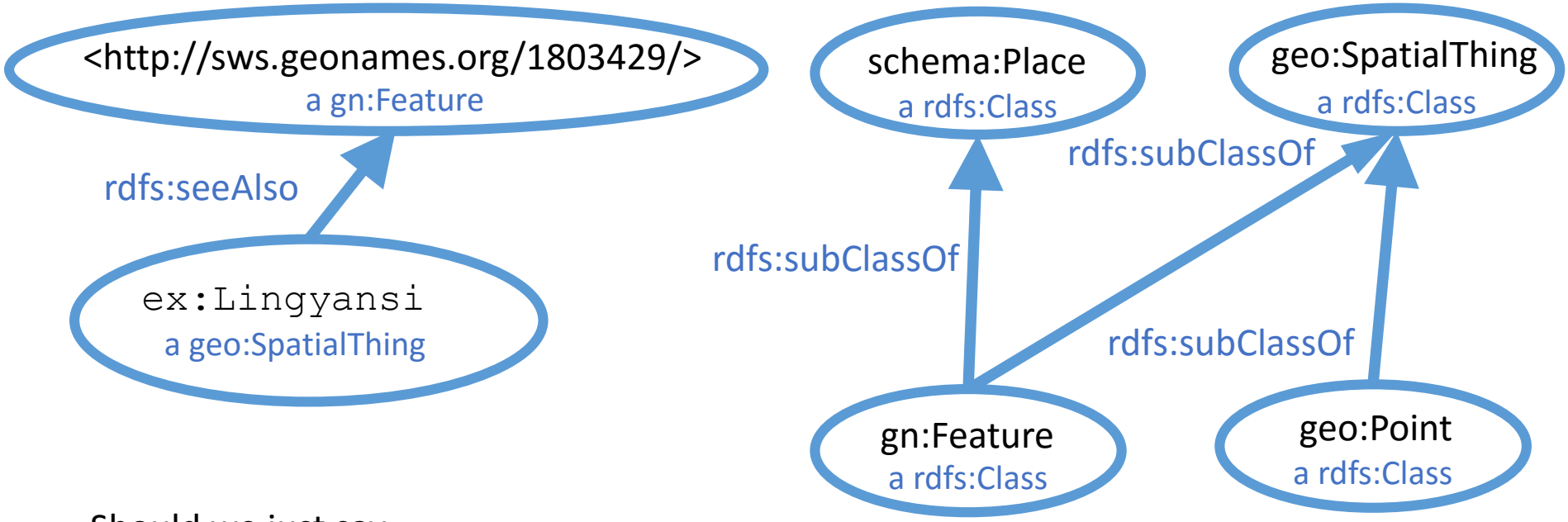
and

```
_:1 a geo:Point.
```

then

```
_:1 a geo:SpatialThing.
```

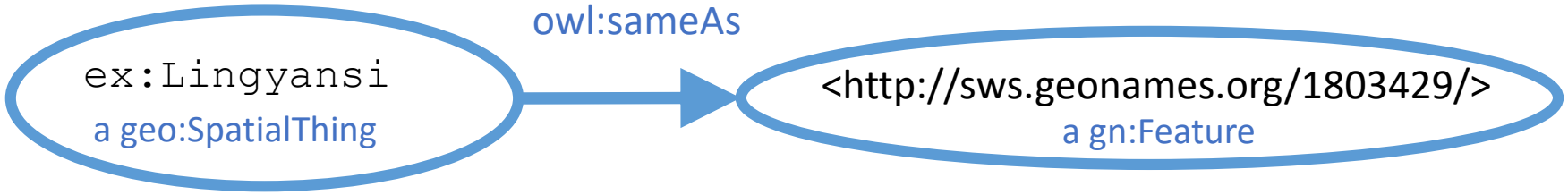
Subclass relationship from
`http://www.geonames.org/ontology`



Should we just say

```
ex:Lingyansi owl:sameAs <http://sws.geonames.org/1803429/>.
```

?



If we say

```
ex:Lingyansi
     rdfs:label "Lingyan Temple "@en;
     dcterms:temporal _:2 ;
     a geo:SpatialThing.

_:2  rdf:value "Tang Dynasty to Ching Dynasty"@en;
     a dcterms:PeriodOfTime.
```

and we say

```
ex:Lingyansi owl:sameAs <http://sws.geonames.org/1803429/>.
```

then

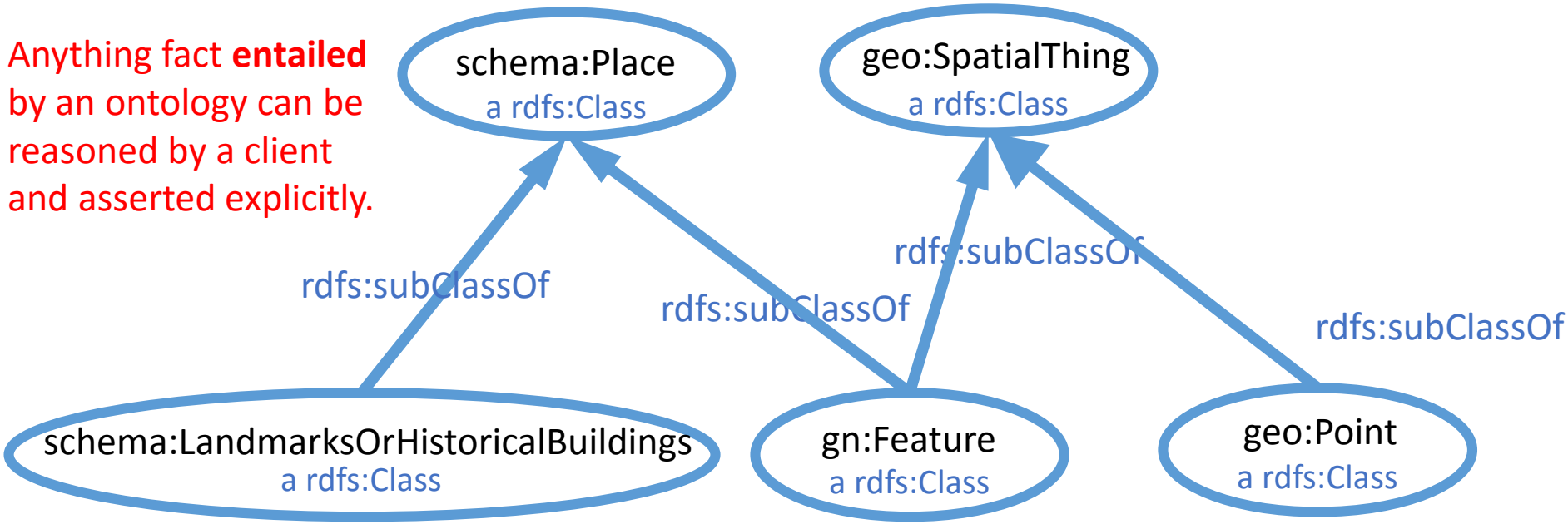
```
<http://sws.geonames.org/1803429/>
     rdfs:label "Lingyan Temple "@en;
     dcterms:temporal _:2 ;
     a geo:SpatialThing.

_:2  rdf:value "Tang Dynasty to Ching Dynasty"@en;
     a dcterms:PeriodOfTime.
```

Everything asserted about one resource is also asserted about the other resource = **dangerous**!

Explaining relationships about what things **ARE** is called an **ontology**.

Anything fact **entailed** by an ontology can be reasoned by a client and asserted explicitly.



If

```
<http://sws.geonames.org/1803429/> a gn:Feature.
```

then

```
<http://sws.geonames.org/1803429/> a schema:Place.
```

and

```
<http://sws.geonames.org/1803429/> a geo:SpatialThing.
```

Relationships from
`http://vocab.getty.edu/aat/` (Getty Art and Architecture Thesaurus; `att:`)

Explaining relationships about **how we categorize things** is called an **concept scheme**.

SKOS (Simple Knowledge Organization System) is a W3C standard for describing **concept schemes**. SKOS is also very much about controlling labels.

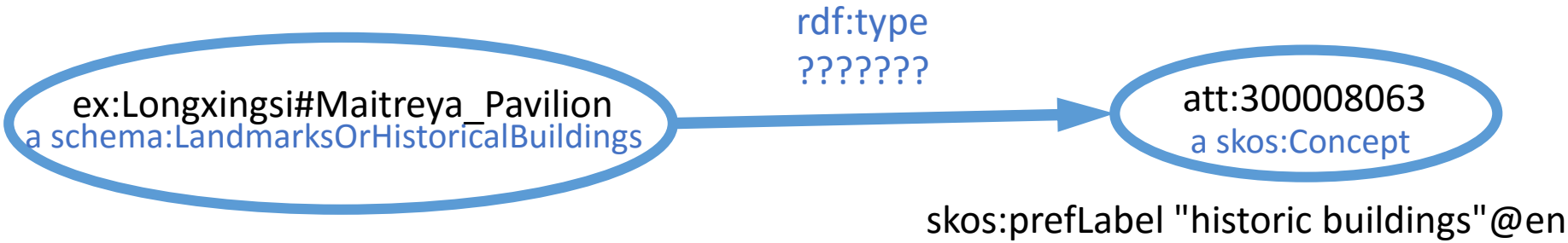
We said:

```
ex:Longxingsi#Maitreya_Pavilion a schema:LandmarksOrHistoricalBuildings.
```

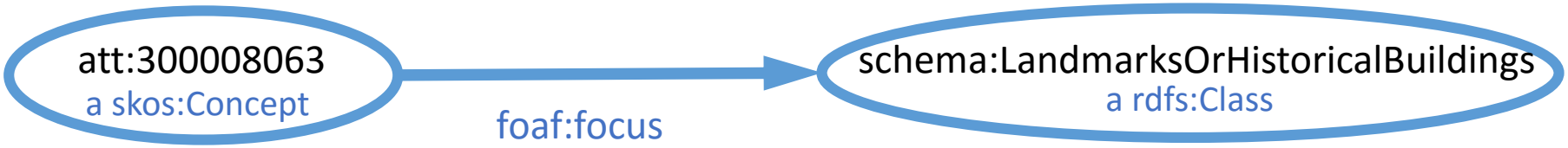
Should we say:

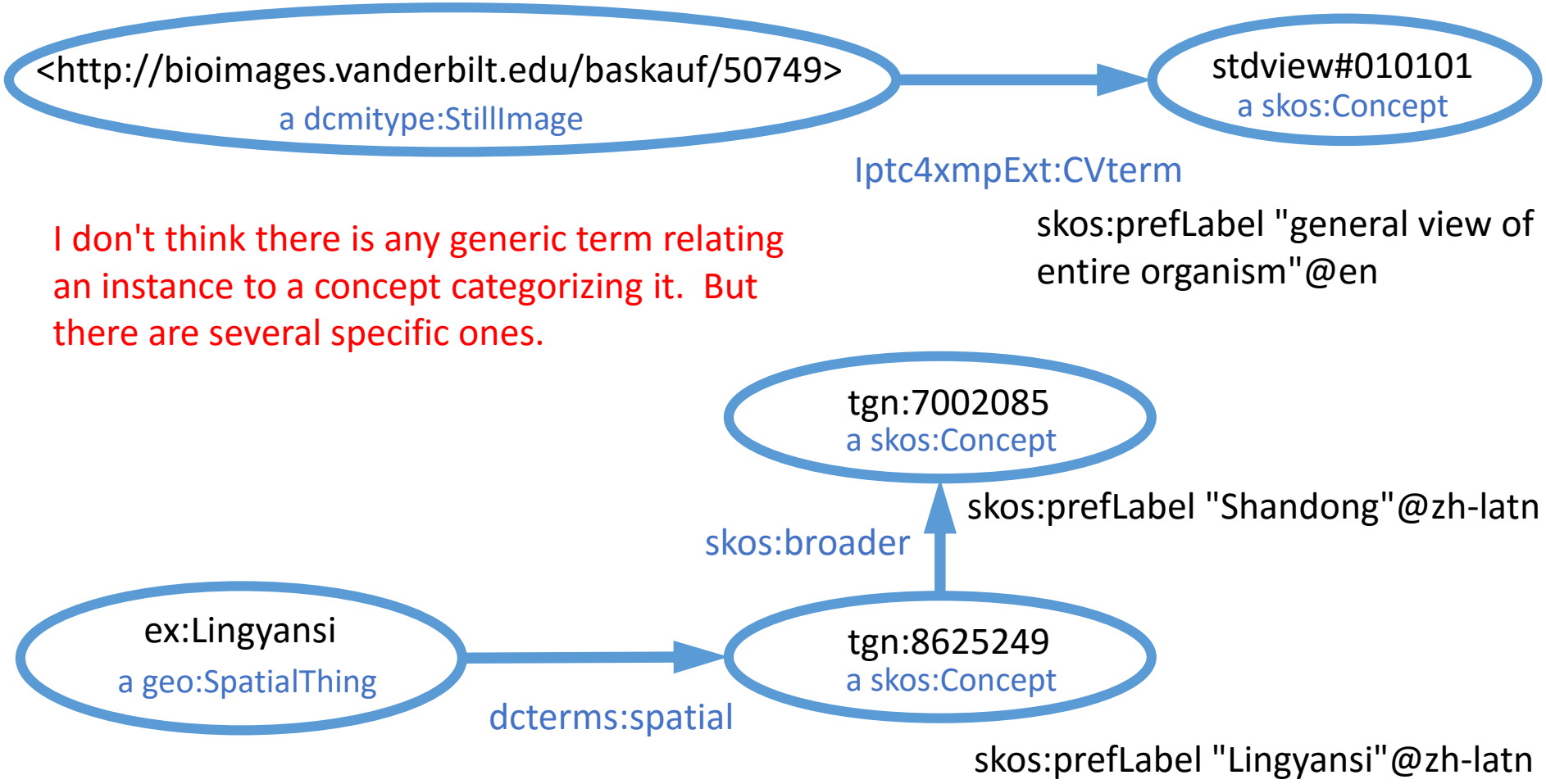
```
ex:Longxingsi#Maitreya_Pavilion a att:300008063.
```

?



The object of an `rdf:type` triple is a class (what the thing is), whereas a `skos:Concept` is "an idea or notion; a unit of thought", and is used to organize and categorize knowledge. Using a `skos:Concept` as a class is consistent with the SKOS data model, but probably not a good idea.





I don't think there is any generic term relating an instance to a concept categorizing it. But there are several specific ones.

If we say:

```
ex:Longxingsi dcterms:spatial tgn:8625249.
```

Does that entail:

```
ex:Longxingsi dcterms:spatial tgn:7002085.
```

? No. A client may lead a human searcher to the broader category, but it's not entailed.

A client programmed to process SKOS can apply various rules from the SKOS specification. But nothing is automatically entailed as with ontology reasoning.

What do people care about?

- **Machine-readable data** people (e.g. Schema.org, RDFa, Microformats) care about making it easier for bots to harvest data from web pages. They care about community vocabularies.
- **Linked Data** people care about linking resources in different silos. They care about IRIs to cross domains and about community vocabularies.
- **Semantic Web** people care about reasoning entailed triples based on ontologies, about IRIs, and about community vocabularies.

What should we care about ????

Next time

- I’m going to try to Skype in. Who can finish Ch. 2?
- I challenge thee to set up your own CSV metadata files to generate RDF triples. See tools to validate and make graphs.
- Section 2.4 is about serializations of RDF
- I challenge thee to do something with the sample files. XML people should try to use RDF/XML, web developers should try to use RDF/JSON. Can anybody harvest RDFa from an enhanced web page???
- Sections 2.5 and 2.6 are about serving content.
- Use RawGit to get the Content-Type right. Use Postman, Advanced Rest Client for Chrome (Windows), or cURL to test.