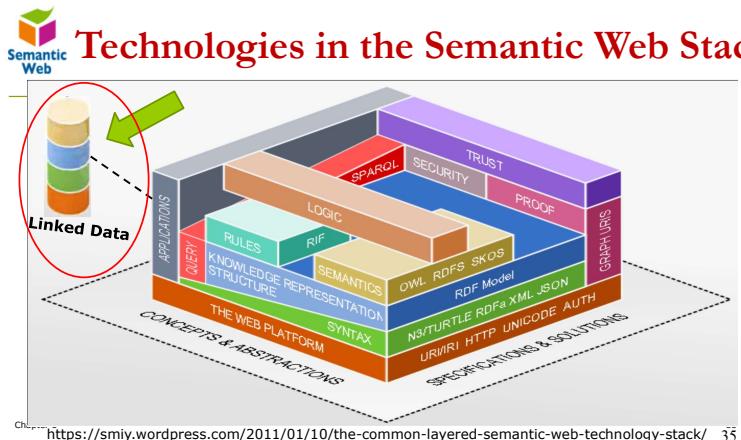


Linked Open Data (LOD)



What is Linked Open Data?

- **Linked Data** คือการเชื่อมโยงข้อมูลต่าง ๆ ที่มีโครงสร้างเข้าด้วยกัน ซึ่งไม่เพียงแต่มนุษย์เท่านั้นที่เข้าใจได้ แต่ยังรวมถึง computer ด้วย
 - ถ้าข้อมูลที่มีการเชื่อมโยงนั้นเป็นที่普遍เผยแพร่กับสาธารณะ เราจะ เรียกว่า **Linked Open Data** หรือ LOD
 - LOD จะเป็นพื้นฐานของ Semantic Web



Technologies in the Semantic Web Stack



What is Linked Open Data?

- LOD ដើម្បីការងារ
ទីផ្សារ [Web of Data](#)
 - LOD គឺទូទៅខ្លួនអូល
ខាងតាំងរឿងក្នុង [URI](#)
ដើម្បីទីនេះជាមួយខ្លួន
ចាប់ពីរបៀប
 - សំវាតែងសាន្តតិចតិចតារាង
បើជាអ្នកប្រើប្រាស់នៃការ
ការងារ [map](#) ឬវិធីផ្ទេរ
ទូទៅខ្លួនដែលត្រូវបាន
បង្កើតឡើង
 - LOD ដើម្បីការងារ
បានបង្កើតឡើង
 - ខ្លួនគ្នានៅទីនេះត្រូវ
ដើម្បីទីនេះជាមួយ

The Linked Open Data cloud of data sets visualizing contents of the Semantic Web.
Modified after Cyganiak and Jentzsch (2010).



Four Principles of Linked Data

- ❑ **Four principles of linked data** (proposed by Tim Berners-Lee for preparing linked data for the semantic web)

1. LOD ต้องใช้ URI หรือ IRI แทนชื่อของสิ่งต่าง ๆ การใช้ URI นี้ทำให้เราสามารถแยกแยะระหว่างสิ่งต่าง ๆ หรือรู้ว่าสิ่งหนึ่งจากชุดข้อมูลหนึ่งเมื่อันหรือแตกต่างจากอีกสิ่งหนึ่งในชุดข้อมูลอื่นได้



Four Principles of Linked Data

2. LOD ยังคงมีการใช้โปรโตคอล **HTTP** ใน URI (เหมือนกับ "URL") เพื่อให้ผู้คนสามารถค้นหาข้อมูลได้ เมื่อจากไปร์โตรถคอล HTTP มีกลไกง่ายๆ ใน การดึง resource บนเว็บ เมื่อสิ่งต่างๆ สามารถระบุได้ด้วย URI ร่วมกับ โปรโตคอลนี้ สิ่งต่างๆ ก็จะค้นหาได้ง่ายขึ้น

Four Principles of Linked Data

3. เมื่อมีคนค้นหาเกี่ยวกับ **URI** ตัว LOD ก็จะให้ข้อมูลที่มีการอธิบายเกี่ยวกับ URI นั้นข้อมูลเกี่ยวกับ URI ก็จะถูกเก็บอยู่ในโครงสร้าง **RDF** นอกจากนี้ LOD ก็ยังเปิดโอกาสให้มีการใช้ **SPARQL** ในการสืบค้นข้อมูลจาก LOD ได้ด้วย ซึ่งข้อมูลที่ถูกเชื่อมโยงกันนี้ ก็จะสามารถถูกอ่านได้ด้วย computer เพราะมีการจัดเก็บในรูปแบบของ **RDF**

39

Four Principles of Linked Data

4. การใช้ URI จะทำให้เราสามารถคลิกที่ **URI** เพื่อค้นพบข้อมูลต่างๆของ URI นั้นเพิ่มเติม เช่นเดียวกับการคลิก **link** ในเอกสาร **WWW** แต่ **link** ใน **WWW** เป็น **link** ที่เชื่อมไปยังเอกสารอื่น (**Document-to-Document**) แต่ **link** ของ **URI** จะเป็น **link** ที่เชื่อมโยงไปยัง **Data** (**Data-to-Data**) นอกจากนี้ LOD ยังช่วยให้เราสามารถเชื่อมโยง **Data** ที่เป็น **resource** ใหม่เข้ากับ **resource** เก่าที่มีอยู่แล้ว หรือเป็นการ reuse **resource** ที่มีคนสร้างไว้แล้วมาใช้กับ **resource** ของเรามาเพิ่มเติมเพื่อให้เกิดประโยชน์สูงสุด

40

Linked Data Vocabularies

- ในปัจจุบันจะมีกลุ่มคำศัพท์มาตรฐาน (**Standard Vocabularies**) ที่ถูกสร้างโดยองค์กรต่างๆ มากมายที่เราสามารถหยิบมายัง **URI/IRI** ของเราได้
- โดยความสามารถสร้าง **URI** ในเอกสาร **RDF** ได้หากหลักวิธีต่อไปนี้:
 1. ใช้คำศัพท์หรือ **vocabulary** ที่เราสร้างขึ้นมาเองมาสร้างเป็น **URI**
 2. ใช้คำศัพท์มาตรฐานที่มีอยู่แล้วที่ถูกสร้างโดยองค์กรต่างๆ มาสร้างเป็น **URI**
 3. ใช้คำศัพท์ที่เราสร้างเอง หรือใช่วร่วมกับคำศัพท์มาตรฐานของที่อื่น

From Wikipedia to DBpedia

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

- Wikipedia เป็นสารานุกรมขนาดใหญ่ ผู้คนในโลกสามารถแบ่งปันความรู้ รวมกัน

- Wikipedia ให้ข้อมูลในรูปแบบข้อความที่ไม่มีโครงสร้างและในรูปแบบตาราง (ข้อมูลโครงสร้าง)
- ข้อมูลที่มีโครงสร้างนี้สามารถแปลงเป็นเอกสาร **RDF** เพื่อใช้ใน DBpedia ได้อย่างง่ายดาย



43

Example of using existing LOD Vocabulary

- ❑ **DBpedia** - ("DB" คือ "Database") เป็นกลุ่มคำศัพท์มาตรฐานที่ถูกสร้างจากการ **Wikipedia** โดยมีจุดมุ่งหมายเพื่อทำให้ข้อมูล (**text**) ใน **Wikipedia** ที่อยู่ในรูปแบบที่ไม่มีโครงสร้าง (**unstructured data**) ทำให้อยู่ในรูปแบบที่มีโครงสร้าง (**structured data**) และสามารถเผยแพร่ออนไลน์ใน **WWW** ได้
- ❑ **DBpedia** อนุญาตให้ผู้ใช้ค้นหา **URI**, **properties** ของ **URL** และค่า **ข้อมูล**ของ **property** แต่ละตัว รวมถึงการเชื่อมโยงไปยัง **URI** ตัวอื่นที่เกี่ยวข้องได้

42

How to search a term in DBpedia?

lookup.dbpedia.org/index.html <https://lookup.dbpedia.org/index.html>

DBpedia Lookup

Documentation at <https://github.com/dbpedia/lookup>
Search API at <https://lookup.dbpedia.org/api/search?query=Pluto>
Auto-Complete API at <https://lookup.dbpedia.org/api/complete?query=Pluto>

Test it here:

Top 10 Results:

- Pluto - <http://dbpedia.org/resource/Pluto>
- Pluto (Disney) - [http://dbpedia.org/resource/Pluto_\(Disney\)](http://dbpedia.org/resource/Pluto_(Disney))
- Moons of Pluto - http://dbpedia.org/resource/Moons_of_Pluto
- List of geological features on Pluto - http://dbpedia.org/resource/List_of_geological_features_on_Pluto
- Pluto Press - http://dbpedia.org/resource/Pluto_Press
- Pluto Satellites - http://dbpedia.org/resource/Pluto_Satellites
- Pluto (mythology) - [http://dbpedia.org/resource/Pluto_\(mythology\)](http://dbpedia.org/resource/Pluto_(mythology))
- Sailor Pluto - http://dbpedia.org/resource/Sailor_Pluto
- Operation Pluto - http://dbpedia.org/resource/Operation_Pluto
- Club Atlético River Plate - http://dbpedia.org/resource/Club_Atlético_River_Plate

• จากเว็บนี้มีรายละเอียด (**Pluto**) สำหรับการค้นหา ผลลัพธ์ของคำว่า "Pluto" จะแสดงในรูปแบบของ **URI resource**

• จากนั้นเราจะสามารถคลิกที่ **URI resource** ที่เราต้องการ เพื่อดึงข้อมูลเพิ่มเติมเกี่ยวกับคำว่า "Pluto" (ดู slide ถัดไป)

44

DBpedia- Example of Web of Data

Using DBpedia Vocabulary in URI

The subject is represented in URI format.

<<https://dbpedia.org/resource/Pluto>>

Stand for



Plut

Real world Object

The property is also represented in URI format.

ดังนั้น หากเราต้องการอ้างถึง "Pluto" ซึ่งเป็นดาวเคราะห์ (Planet) ที่ DBpedia เรายังสามารถที่จะเลือกค่าพาร์ทจาก DBpedia เพื่อแสดง results ของ Dbpedia ที่อยู่ในรูปแบบของ URI เช่นกัน

Example of existing LOD Vocabularies

 Schema.org

- นอกจาก DBpedia แล้ว ยังมีกิจลุ่มคำศัพท์มาตรฐานอื่นๆอีก ที่เสนอโดยองค์กรอื่นๆ เช่น [Schema.org](#)
 - [Schema.org](#) เป็นความคิดริเริ่มที่เริ่มต้นโดย Bing, Google, Yahoo! เมื่อวันที่ 2 มิถุนายน 2554
 - **Goal:** เพื่อนำคำศัพท์เหล่านี้มาใช้สร้างเป็น URI หรือใช้ร่วมอยู่ใน Mark up ของเอกสาร HTML เพื่อใช้อธิบายความหมายสำหรับเนื้อหาในเว็บ
 - ซึ่ง Search engine และเบราว์เซอร์จะจับและเข้าใจ Mark up ที่มีคำศัพท์ของ Schema.org นั้น
 - นอกเหนือไปจากนี้ เอกสาร HTML ที่มีการใช้คำศัพท์จาก schema.org นี้ จะสามารถแปลงให้อยู่ในรูปแบบของ RDF model ในลักษณะของ triple หรือ graph ได้ด้วย

Example of Schema.org Vocabulary

<https://schema.org/Person>

The screenshot shows the Schema.org Person page with several annotations:

- A green arrow points from the "URI resource" label at the top right down to the "Properties from Person" table.
- A red circle highlights the "Properties from Person" section.
- Red arrows point from the "Additional Name" and "Physical Address" rows to their respective definitions below.
- A blue arrow points from the "Affiliation" row to its definition below.
- A green arrow points from the "Educational Organization" row to its definition below.
- A red arrow points from the "Alumni Of" row to its definition below.
- A blue arrow points from the "Award" row to its definition below.
- A red arrow points from the "Birth Date" row to its definition below.
- A green arrow points from the "Birth Place" row to its definition below.
- A red arrow points from the "Brand" row to its definition below.
- A blue arrow points from the "Callsign" row to its definition below.
- A red arrow points from the "Children" row to its definition below.

Property	Expected Type	Description
additionalName	Text	An additional name for a Person, can be used for a middle name. Physical address of the item.
address	Address or Text	
affiliation	Organization	An organization that this Person is affiliated with. For example, a school or a club, or a team.
alumniOf	EducationalOrganization or Organization	An organization that the person is an alumnus of. Inverse property: alumni.
award	Text	An award won for this item. Supersedes awards.
birthDate	Date	Date of birth.
birthPlace	Place	The place where the person was born.
brand	Brand or Organization	The brand(s) associated with a product or service, or the brand(s) maintained by an entity or business person.
callsign	Text	A callsign for shortwave and radio communications to identify people, radio and TV stations, or vehicles.
children	Person	A child of the person.
...	Person or Text	A colleague of the person. Supersedes colleagues.

- หากเราใส่ URI ใน schema.org เป็น "https://schema.org/Person" schema.org จะให้ข้อมูลเกี่ยวกับ Properties ต่างๆ ของ Person ที่ เมื่อเรียบเบนกับบันทึกกลุ่มคำศัพท์ที่ เรากำหนดไว้ในไฟล์เพื่อประกอบการ อธิบายเงื่อนไข (person) ที่เรากำลัง พูดถึงอยู่ในเอกสาร HTML
 - Properties เหล่านี้ก็จะมีลักษณะเป็น URI ที่ความสามารถคิดให้เราไปถูก รายละเอียดอื่นได้ และถูกใช้เป็น คำศัพท์มาตรฐานใน Schema.org

Example of Schema.org Vocabularies

The diagram illustrates the relationship between two Schema.org types: Person and address.

schema.org/Person (highlighted in red) is a Schema.org type representing a Thing > Person. It includes properties like `name`, `address`, `affiliation`, `agentInteractionStatistic`, and `alumniOf`.

schema.org/address (highlighted in green) is another Schema.org type representing a PhysicalAddress. It includes properties like `streetAddress`, `locality`, `region`, `postCode`, and `countryName`.

A green arrow points from the `address` property of the Person schema to the address schema itself, indicating that a Person can have an address.

Another URI resource is shown in a yellow box with the URL <https://schema.org/address>.

แต่ละ Property จะเป็น URI ที่เราสามารถคลิกเพื่อดูรายละเอียดของ URI แต่ละรายการต่อไปได้

Another Example of resource “Place” in schema.org

The screenshot shows the Schema.org Place page with a red circle highlighting the 'additionalProperty' property in the table. A green arrow points from the text 'Properties ของ Place ที่เราสามารถนำไปใช้ในรูปแบบของ URI หรือใช้ประกอบกับ Mark up ใน HTML เพื่อยืนยันข้อมูลสถานที่ที่อยู่ในเอกสาร HTML' to the 'additionalProperty' row in the table.

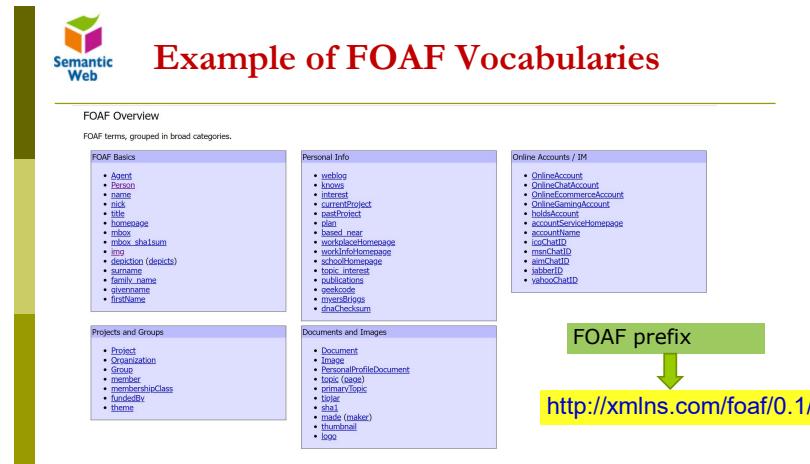
Property	Expected Type	Description
Properties from Place	PropertyValue	A property-value pair representing an additional characteristic of the entity, e.g. a product feature or another characteristic for which there is no matching property in schema.org.
additionalProperty	PropertyValue	Note: Publishers should be aware that applications designed to use specific schema.org properties (e.g. https://schema.org/width , https://schema.org/availableAt , etc.) may ignore values for this property.
address	PostalAddress	
aggregateRating	Text	
amenityFeature	AggregateRating	
	LocationFeatureSpecification	
	Text	

Properties ของ Place ที่เราสามารถนำไปใช้ในรูปแบบของ URI หรือใช้ประกอบกับ Mark up ใน HTML เพื่อยืนยันข้อมูลสถานที่ที่อยู่ในเอกสาร HTML

Example of Linked Data Vocabularies

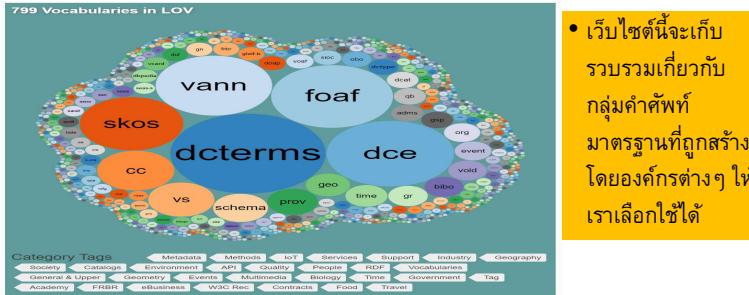
FOAF- Friend of a Friend

- เป็นคำศัพท์ยอดนิยมอีกกลุ่มคำศัพท์หนึ่งที่นิยมใช้ในการอธิบายเกี่ยวกับบุคคล กิจกรรม ความสัมพันธ์กับบุคคลและสิ่งของต่างๆ
 - More information about FOAF
→ <http://xmlns.com/foaf/0.1>



Where to find Linked Data Vocabularies?

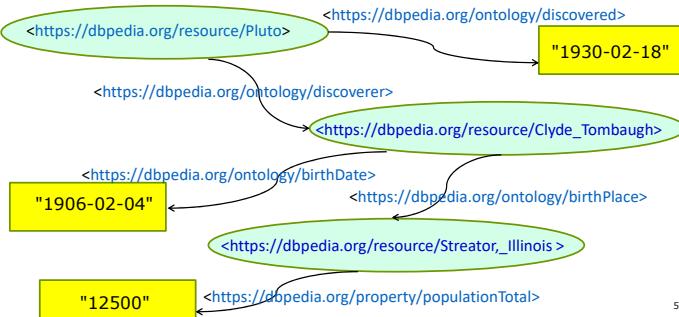
**Linked Open
Vocabularies** → <https://lov.linkeddata.es/dataset/lov/>



- เว็บไซต์นี้เก็บ
รวมรวมเกี่ยวกับ
กลุ่มคำศัพท์
มาตรฐานที่ถูกสร้าง
โดยองค์กรต่างๆ ให้
เราเลือกใช้ได้



This Example represents “Pluto” by using DBpedia Vocabulary in Graph Format



5



Subject	Property	Object/Value
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discoveredBy>	"1930-02-18"^^xsd:date
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discoveredBy>	<https://dbpedia.org/resource/Clyde_Tombaugh>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/CelestialBody>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/Planet>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://schema.org/Place>
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthDate>	"1906-02-04"^^xsd:date
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthPlace>	<https://dbpedia.org/resource/Streator,_Illinois>
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/property/populationTotal>	"12500"^^xsd:integer
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/motto>	"Quiet Surprise on the Prairie"
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/postalCode>	"61364"^^xsd:integer



Representing resource in DBpedia as Triple

- Resources หรือ Instances จะถูกวางอยู่ในส่วนที่ท้าทันที่เป็น Subject หรือ Object ก็ได้

Subject	Property	Object/Value
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discovered>	"1930-02-18"^^xsd:date
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discoverer>	<https://dbpedia.org/resource/Clyde_Tombaugh>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/CelestialBody>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/Planet>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://schema.org/Place>
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthDate>	"1906-02-04"^^xsd:date
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthPlace>	<https://dbpedia.org/resource/Streator,_Illinois>
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/property/populationTotal>	"12500"^^xsd:integer
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/motto>	"Quiet Surprise on the Prairie"
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/postalCode>	"61364"^^xsd:integer



Representing resource in DBpedia as Triple

- Classes จะถูกวางไว้ในส่วนที่เป็น Object อยู่หลัง property ที่ชื่อว่า <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>

Subject	Property	Object/Value
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discovered>	"1930-02-18"^^xsd:date
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discoverer>	<https://dbpedia.org/resource/Clyde_Tombaugh>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/CelestialBody>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/Planet>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://schema.org/Place>
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthDate>	"1906-02-04"^^xsd:date
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthPlace>	<https://dbpedia.org/resource/Streator,_Illinois>
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/property/populationTotal>	"12500"^^xsd:integer
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/motto>	"Quiet Surprise on the Prairie"
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/postalCode>	"61364"^^xsd:integer



Representing resource in DBpedia as Triple

- Literal เป็นค่าข้อมูลของ Property ที่สามารถมีค่าเป็น String, Integer, Date/time, ฯลฯ ได้

Subject	Property	Object/Value
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discovered>	"1930-02-18"^^xsd:date
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discoverer>	<https://dbpedia.org/resource/Clyde_Tombaugh>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/CelestialBody>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/Planet>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://schema.org/Place>
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthDate>	"1906-02-04"^^xsd:date
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthPlace>	<https://dbpedia.org/resource/Streator,_Illinois>
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/property/populationTotal>	"12500"^^xsd:integer
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/motto>	"Quiet Surprise on the Prairie"
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/postalCode>	"61364"^^xsd:integer



Representing resource in DBpedia as Triple

- Property ก็ต้องเขียนให้อยู่ในรูปแบบของ URI ที่สามารถคลิกเข้าไปดูรายละเอียดของ Properties เหล่านี้ได้ด้วย

Subject	Property	Object/Value
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discovered>	"1930-02-18"^^xsd:date
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discoverer>	<https://dbpedia.org/resource/Clyde_Tombaugh>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/CelestialBody>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/Planet>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://schema.org/Place>
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthDate>	"1906-02-04"^^xsd:date
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthPlace>	<https://dbpedia.org/resource/Streator,_Illinois>
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/property/populationTotal>	"12500"^^xsd:integer
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/motto>	"Quiet Surprise on the Prairie"
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/postalCode>	"61364"^^xsd:integer



Representing resource in DBpedia as Triple

- URI สามารถสร้างขึ้นมาโดยเลือกใช้กลุ่มคำศัพท์จากองค์กรที่หลากหลาย เช่น จักรยานิรภัย ที่ใช้ URI จาก dbpedia, w3c และ schmea.org
- บางองค์กร เช่น w3.org ใช้ "#" เป็นตัวแทนภายใน URI

Subject	Property	Object/Value
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discovered>	"1930-02-18"^^xsd:date
<https://dbpedia.org/resource/Pluto>	<https://dbpedia.org/ontology/discoverer>	<https://dbpedia.org/resource/Clyde_Tombaugh>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/CelestialBody>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://dbpedia.org/ontology/Planet>
<https://dbpedia.org/resource/Pluto>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://schema.org/Place>
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthDate>	"1906-02-04"^^xsd:date
<https://dbpedia.org/resource/Clyde_Tombaugh>	<https://dbpedia.org/ontology/birthPlace>	<https://dbpedia.org/resource/Streator,_Illinois>
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/property/populationTotal>	"12500"^^xsd:integer
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/motto>	"Quiet Surprise on the Prairie"
<https://dbpedia.org/resource/Streator,_Illinois>	<https://dbpedia.org/ontology/postalCode>	"61364"^^xsd:integer



Example of using FOAF in RDF triple

- บาง URI เราก็อาจสร้างขึ้นมาเองได้เลย เช่น <http://mydomain.com/Robert> แต่บาง URI เราจะไปยืมมาจากขององค์กรต่างๆ ได้ เช่น ตัวนี้ใช้ URI จาก FOAF และ w3c

Subject	Property	Object/Value
<http://mydomain.com/Robert>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://xmlns.com/foaf/0.1/Person>
<http://mydomain.com/Robert>	<http://xmlns.com/foaf/0.1/name>	"Robert Lee"
<http://mydomain.com/Robert>	<http://xmlns.com/foaf/0.1/mbox>	<mailto:robert@gmail.com>
<http://mydomain.com/Robert>	<http://xmlns.com/foaf/0.1/homepage>	<http://www.robert.com>
<http://mydomain.com/Robert>	<http://xmlns.com/foaf/0.1/img>	<http://www.robert.com/pics/robert.jpg>
<http://mydomain.com/Robert>	<http://xmlns.com/foaf/0.1/interest>	<http://linkeddata.org>
<http://mydomain.com/Robert>	<http://xmlns.com/foaf/0.1/knows>	<http://otherdomain.com/Mary>
<http://mydomain.com/Robert>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://xmlns.com/foaf/0.1/Person>
<http://mydomain.com/Robert>	<http://xmlns.com/foaf/0.1/name>	"Mary Jane"

RDF as the Serialization Formats



RDF as the Serialization Formats

- Graph จะเป็นรูปแบบการแทนโครงสร้างข้อมูลที่มีนุชย์เข้าใจได้ง่าย
- แต่จุดประสงค์ของ Semantic Web คือต้องการที่จะทำให้ computers สามารถเข้าใจความหมายของข้อมูล นำข้อมูลไปประมวลผลได้อย่างถูกต้อง และสืบค้นข้อมูลได้ตรงกับความต้องการของมนุษย์ให้มากที่สุด
- ดังนั้น จึงต้องมีการแทนโครงสร้าง RDF ให้อ่ายံในรูปแบบที่ computers สามารถเข้าใจได้คืออยู่ในรูปแบบของ **RDF Serialization Languages** ได้แก่ภาษา **XML, N3, N-Triples, Turtle, RDFA และ JSON-LD** เป็นต้น

5



RDF as the Serialization Formats

- RDF ไม่ใช้แค่เป็นการจัดรูปแบบของข้อมูล (data format) แต่จะเป็นการสร้างโมเดล ข้อมูล (data model) เพื่อให้สามารถอธิบาย resources ต่างๆ ให้อยู่ในรูปแบบของ Triple ซึ่งได้แก่ **subject, predicate, object**
- จะมีรูปแบบของภาษาที่ถูกเรียกว่า **RDF serialization languages** ได้ถูกนำเสนอขึ้นมา เพื่อให้สามารถนำไปแปลงเป็น **RDF model** คืออยู่ในรูปแบบที่เป็น **triple** ได้ ได้แก่
ภาษาต่อไปนี้
 - **RDF/XML**
 - **Turtle** (Terse RDF Triple Language)
 - **N-Triples**
 - **N-Quads**
 - **RDFA** (Resource Description Framework in Attributes)
 - **RDF/JSON or JSON-LD** (JSON-Linked Data)

RDF/XML

6



RDF/XML

- **RDF/XML** เป็นภาษาที่เน้นโครงสร้างไวยากรณ์ (syntax) ซึ่งมีองค์กร **W3.org** เป็นผู้กำหนดรูปแบบโครงสร้าง โดยเป็นภาษาที่ถูกใช้เป็นภาษาแรกสุดในกลุ่มของ **Serialization Languages** ที่ถูกใช้ในการแทน RDF Model



DB → XML

- เพื่อให้เข้าใจโครงสร้างภาษา XML จะเปลี่ยนเทียบกับโครงสร้างของตารางใน Relational Database ดังต่อไปนี้
- จากตาราง **Planet** ให้ทำการแปลงโครงสร้างข้อมูลในตารางนี้ให้อยู่ในรูปแบบของเอกสาร **XML**

Planet

Convert (แปลง)	ID	mpcName	discovered
	https://dbpedia.org/resource/Pluto	Pluto (en)	1930-02-18

```
<?xml version="1.0"?>
<Planet id="https://dbpedia.org/resource/Pluto">
  <mpcName>Pluto (en)</mpcName>
  <discovered>1930-02-18</discovered>
</Planet>
```

ElementAttribute

8

9



Insert Namespace to XML Element

- ตัดไปก่อการประกาศ **prefixes** หรือที่เรียกว่า **XML namespaces (xmlns)** เพื่อให้ไปประหน้า elements ทุกด้วยเพื่อทำให้ elements ทุกด้วยมีความเป็นหนึ่งเดียวคือไม่ซ้ำกัน (unique)

```
<?xml version="1.0"?>
<Planet id="https://dbpedia.org/resource/Pluto">
  <mpcName>Pluto (en)</mpcName>
  <discovered>1930-02-18</discovered>
</Planet>
```

Planet.xml

```
<?xml version="1.0"?>
<dbo:Planet id="https://dbpedia.org/resource/Pluto"
  xmlns:dbo="https://dbpedia.org/ontology"
  xmlns:dbp="https://dbpedia.org/property">
  <dbp:mpcName>Pluto (en)</dbp:mpcName>
  <dbp:discovered>1930-02-18</dbp:discovered>
</dbo:Planet>
```

Planet.xml

เขียน **dbo:Planet** จะถูกแปลความหมายเป็น "<https://dbpedia.org/ontology/Planet>"

กำหนดให้ตัวบปร **dbo** เป็น Prefix หรือ **xmlns** ที่มีค่า เท่ากับ "<https://dbpedia.org/ontology>" และ **dbp** เป็น prefix ที่มีค่า เท่ากับ "<https://dbpedia.org/property>" และให้มา prefix **dbo**, **dbp** ไป ประหน้า Elements แต่ละตัว



XML → RDF

<https://www.w3.org/RDF/Validator/>

- ปรับรูปแบบเอกสาร XML ต่อไปนี้เพื่อทำให้อยู่ในรูปแบบเอกสาร RDF ที่ถูกต้อง:

```
<?xml version="1.0"?>
<dbo:Planet id="https://dbpedia.org/resource/Pluto"
  xmlns:dbo="https://dbpedia.org/ontology"
  xmlns:dbp="https://dbpedia.org/property">
  <dbp:mpcName>Pluto (en)</dbp:mpcName>
  <dbp:discovered>1930-02-18</dbp:discovered>
</dbo:Planet>
```

XML

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="https://dbpedia.org/resource/Pluto"
    xmlns:dbp="https://dbpedia.org/property/">
    <dbp:type rdf:resource="https://dbpedia.org/ontology/Planet" />
    <dbp:mpcName>Pluto (en)</dbp:mpcName>
    <dbp:discovered>1930-02-18</dbp:discovered>
  </rdf:Description>
</rdf:RDF>
```

RDF

convert to

Planet.xml

Planet.rdf

11



Triple of the Data Model

Validation Results

<https://www.w3.org/RDF/Validator/>

Your RDF document validated successfully.

Triples of the Data Model

Number	Subject	Predicate	Object
1	https://dbpedia.org/resource/Pluto	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	https://dbpedia.org/ontology/Planet
2	https://dbpedia.org/resource/Pluto	https://dbpedia.org/property/mpcName	"Pluto (en)"
3	https://dbpedia.org/resource/Pluto	https://dbpedia.org/property/discovered	"1930-02-18"

ผลลัพธ์ที่ได้จาก RDF validator โดยแสดงผลในรูปแบบของ Triple

12



Graph of the Data Model

Graph of the data model

<https://www.w3.org/RDF/Validator/>



ผลลัพธ์ที่ได้จาก RDF validator โดยแสดงผลในรูปแบบของ Graph

13

Turtle- Terse RDF Triple Language



Turtle- Terse RDF Triple Language

- Turtle จะเป็นภาษาอิภากภาษาหนึ่งที่ทำให้เราสามารถแทน RDF graphs ให้อยู่ในรูปแบบของภาษาที่ใกล้เคียงกับภาษาธรรมชาติของมนุษย์ คือ **ประโยคภาษาอังกฤษ**
- Turtle จะต้องมีการประกาศตัวแปรในรูปแบบของ **Prefixes** และใช้ตัวแปรที่เป็น **Prefixes** นั้นเป็น **Resources** เพื่อทำให้อยู่ในรูปแบบของ **URIs** ทำให้ไม่ต้องเขียน URIs ในรูปแบบที่ยาวเกินไป

15

Turtle- Terse RDF Triple Language

Example1 of Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dbp: <https://dbpedia.org/property/>.
@prefix dbo: <https://dbpedia.org/ontology/>.
@prefix dbr: <https://dbpedia.org/resource/>.
```

dbr:Pluto

```
rdf:type dbo:Planet;
dbp:mpcName "Pluto (en)";
dbp:discovered "1930-02-18".
```

เขียนได้รูปแบบดัง

```
a dbo:Planet;
```

ตั้งชื่อไฟล์ชื่น
planet.ttl

การประกาศ prefix ให้กับตัวแปร
แล้วตัวแปรที่มีบัญชีนี้
และต้องสืบค้นด้วยเครื่องหมาย
ทุกบรรทัด

เครื่องหมาย ; และถ้า dbr:Pluto จะถูก^{ใช้ให้เป็นประ ранในประโยคถัดไปอีก}
เครื่องหมาย . แสดงถึงจุดสิ้นสุดของประโยค^{ที่มี dbr:Pluto เป็นประ ранของประโยค}

16

Turtle- Terse RDF Triple Language

Example1 of Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dbp: <https://dbpedia.org/property/>.
@prefix dbo: <https://dbpedia.org/ontology/>.
@prefix dbr: <https://dbpedia.org/resource/>.
```

dbr:Pluto

```
a dbo:Planet;
dbp:mpcName "Pluto (en)";
dbp:discovered "1930-02-18".
```

มีความหมายแบบเดียวกับการเขียนแบบ
triples ดังนี้:
(dbr:Pluto, rdf:type, dbo:Planet)
(dbr:Pluto, dbp:mpcName, "Pluto (en)")
(dbr:Pluto, dbp:discovered, "1930-02-18")

17

Turtle- Terse RDF Triple Language

ทดสอบความถูกต้องของ “Planet.ttl” ได้โดยใช้เครื่องมือต่อไปนี้

<https://www.semantechs.co.uk/turtle-editor-viewer/>

The screenshot shows the Semantechs Turtle Editor-Viewer interface. On the left, the turtle code is displayed:

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
2 @prefix dbp: <https://dbpedia.org/property/>.
3 @prefix dbo: <https://dbpedia.org/ontology/>.
4 @prefix dbr: <https://dbpedia.org/resource/>.
5
6
7 dbr:Pluto
8   rdf:type dbo:Planet;
9   dbp:mpcName "Pluto (en)";
10  dbp:discovered "1930-02-18".
11
12
13
14
15
16
```

On the right, a graph visualization shows nodes for dbr:Pluto, dbo:Planet, Pluto (en), and 1930-02-18, connected by edges for rdf:type, dbp:mpcName, and dbp:discovered respectively.

Turtle- Terse RDF Triple Language

หรือสามารถทดสอบความถูกต้องของ turtle code ได้ที่ <http://ttl.summervofcode.be/>

The screenshot shows the IDLab Turtle Validator interface. The turtle code is pasted into the input field:

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
2 @prefix dbp: <https://dbpedia.org/property/>.
3 @prefix dbo: <https://dbpedia.org/ontology/>.
4 @prefix dbr: <https://dbpedia.org/resource/>.
5
6
7 dbr:Pluto
8   a dbo:Planet;
9   dbp:mpcName "Pluto (en)";
10  dbp:discovered "1930-02-18".
11
12
13
14
15
16
```

Below the code, a message says "Congrats! Your syntax is correct."

Turtle- Terse RDF Triple Language

Example2 of Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dbp: <https://dbpedia.org/property/>.
@prefix dbo: <https://dbpedia.org/ontology/>.
@prefix dbr: <https://dbpedia.org/resource/>.
```

dbr:Pluto

```
a = rdf:type
a dbo:Planet;
dbp:mpcName "Pluto (en)";
dbp:discovered "1930-02-18";
dbp:discoverer dbr:Clyde_Tombaugh.
```

dbr:Clyde_Tombaugh

```
a dbo:Person;
dbo:deathDate "1997-01-17".
```

ตั้งชื่อไฟล์ชื่น
Planet2.ttl

20

Turtle- Terse RDF Triple Language

ทดสอบความถูกต้องของเอกสาร “Planet2.ttl” ได้โดยใช้เครื่องมือ

<https://www.semantechs.co.uk/turtle-editor-viewer/>

The screenshot shows the Semantechs Turtle Editor-Viewer interface. On the left, the turtle code is displayed:

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
2 @prefix dbp: <https://dbpedia.org/property/>.
3 @prefix dbo: <https://dbpedia.org/ontology/>.
4 @prefix dbr: <https://dbpedia.org/resource/>.
5
6
7 dbr:Pluto
8   a = rdf:type
9   a dbo:Planet;
10  dbp:mpcName "Pluto (en)";
11  dbp:discovered "1930-02-18";
12  dbp:discoverer dbr:Clyde_Tombaugh.
13
14
15
16
```

On the right, a graph visualization shows nodes for dbr:Pluto, dbo:Planet, Pluto (en), 1930-02-18, and dbr:Clyde_Tombaugh, connected by edges for a = rdf:type, a dbo:Planet, dbp:mpcName, dbp:discovered, and dbp:discoverer respectively.



Turtle- Terse RDF Triple Language

- ทดสอบความถูกต้องของเอกสาร "Planet2.ttl" โดยใช้เครื่องมือ

<https://www.semantics.co.uk/turtle-editor-viewer/>

The screenshot shows the Turtle Editor-Viewer interface. On the left is the source code of Planet2.ttl, which defines triples for Pluto, Clyde Tombaugh, and Spiderman. On the right is a graph visualization where nodes like 'dbr:Pluto', 'dbr:Clyde_Tombaugh', and 'dbo:Person' are connected by edges labeled 'rdf:type' and 'dbo:deathDate'. A yellow callout box points to the '@base <http://example.org/>' line in the code, explaining its use for defining the base namespace.

```

1 @base <http://example.org/> .
2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
4 @prefix foaf: <http://xmlns.com/foaf/0.1/> .
5 @prefix rel: <http://www.perceive.net/schemas/relationship/> .
6
7 dbr:Pluto
8 a dbo:Planet;
9 dbo:planetName "Pluto (en)" ;
10 dbo:discovered "1930-02-18" ;
11 dbo:discoverer dbr:Clyde_Tombaugh .
12
13 dbr:Clyde_Tombaugh
14 a dbo:Person;
15 dbo:deathDate "1997-01-17" .
16
  
```

Example3 of Turtle

@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

<green-goblin>
rel:enemyOf <spiderman> ;
a foaf:Person ; # in the context of the Marvel universe
foaf:name "Green Goblin".

<spiderman>
rel:enemyOf <green-goblin> ;
a foaf:Person ;
foaf:name "Spiderman", "Uomo Ragno"@it .

A yellow callout box highlights the use of the base URL and the context of the Marvel universe for the properties.



Turtle- Terse RDF Triple Language

- Try validate "Planet2.ttl" with <https://www.semantics.co.uk/turtle-editor-viewer/>

The screenshot shows the validation results for Planet2.ttl. It displays the same triples as the first example, with the graph visualization showing the relationships between the entities. A yellow callout box points to the '@base <http://example.org/>' line, emphasizing its role in defining the base namespace.

```

1 @base <http://example.org/> .
2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
4 @prefix foaf: <http://xmlns.com/foaf/0.1/> .
5 @prefix rel: <http://www.perceive.net/schemas/relationship/> .
6
7
8 <green-goblin>
9 rel:enemyOf <spiderman> ;
10 a foaf:Person ; # in the context of the Marvel universe
11 foaf:name "Green Goblin".
12
13 <spiderman>
14 rel:enemyOf <green-goblin> ;
15 a foaf:Person ;
16 foaf:name "Spiderman", "Uomo Ragno"@it .
17
  
```

Try validate "Planet2.ttl" with <https://www.semantics.co.uk/turtle-editor-viewer/>

The screenshot shows the validation results for Planet2.ttl. It displays the same triples as the first example, with the graph visualization showing the relationships between the entities. A yellow callout box points to the '@base <http://example.org/>' line, emphasizing its role in defining the base namespace.



Turtle- Terse RDF Triple Language

Example3 of Turtle

```

@prefix : <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .
  
```

:green-goblin
rel:enemyOf **:spiderman** ;
a foaf:Person ; # in the context of the Marvel universe
foaf:name "Green Goblin".

:spiderman
rel:enemyOf **:green-goblin** ;
a foaf:Person ;
foaf:name "Spiderman", "Uomo Ragno"@it .

เราสามารถใช้ @prefix : เพื่อแทน @base ได้
ถ้าด้วย namespace : ปะกู้หัว resource ที่
ต้องการได้เลย

Example4 of Turtle

@prefix dbo: <http://dbpedia.org/ontology/> .
@base <http://dbpedia.org/resource/> .

<Pluto> dbo:hasSatellite <Hydra>,
<Nyx>, <Charon> .

A yellow callout box highlights the use of the base URL and the context of the Marvel universe for the properties.

Graph visualization showing the relationships between Pluto, Hydra, Nyx, and Charon.



N-Triples



N-Triples

- ▣ N-Triples จะเป็น subset ของ Turtle โดยจะตัดเรื่องการใช้ namespace prefixes ออกไป
- ▣ ในแต่ละ triple จะต้องมีการระบุ URIs ทุกดัวอย่างเต็มรูปแบบโดยไม่มีการใช้ตัวแปร prefix ที่ทำให้การเขียนสั้นลง
- ▣ ดังนั้นไฟล์ N-Triples อาจมีค่อนข้างใหญ่เมื่อเทียบกับ Turtle และแม้แต่ RDF/XML
- ▣ อย่างไรก็ตาม N-Triples ถูกออกแบบให้มีรูปแบบที่ง่ายกว่า turtle และง่ายต่อ software ที่จะทำการอ่านและแปลงข้อมูลให้ออกในรูปแบบของ triple
- ▣ N-Triples จึงเหมาะสมที่จะใช้ในการแลกเปลี่ยนข้อมูลที่มีขนาดใหญ่ เพื่อใช้ประมวลผล RDF graph ที่มีขนาดใหญ่ด้วย tool หรือ software ที่มีลักษณะการประมวลผลความที่ต้องอ่านเป็นบรรทัดๆ

31



N-Triples

Example of N-Triples

```
<https://dbpedia.org/resource/Pluto> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://dbpedia.org/ontology/Planet> .  
<https://dbpedia.org/resource/Pluto> <https://dbpedia.org/property/discovered> "1930-02-18" .  
<https://dbpedia.org/resource/Pluto> <https://dbpedia.org/propertydiscoverer> <https://dbpedia.org/resource/Clyde_Tombaugh> .  
<https://dbpedia.org/resource/Clyde_Tombaugh> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://dbpedia.org/ontology/Person> .  
<https://dbpedia.org/resource/Clyde_Tombaugh> <https://dbpedia.org/ontology/deathDate> "1997-01-17" .
```

32



N-Triples

ให้ทดลองใช้เครื่องมือที่ในการแปลงเอกสาร RDF/XML หรือ Turtle ให้เป็น N-Triples
<https://www.easyrdf.org/converter>

The screenshot shows the 'easyrdf.org/converter' interface. In the 'Input Data' section, there is a block of Turtle code:

```
@prefix dbo: <http://dbpedia.org/ontology/> .  
@base <http://dbpedia.org/resource/> .  
<Pluto> dbo:hasSatellite <Charon> .  
<Charon> dbo:name <Nyx> .
```

In the 'Input Format' dropdown, 'Turtle/Turtle ROF Triple Language' is selected. In the 'Output Format' dropdown, 'N-Triples' is selected. A yellow callout box points to this selection with the text 'ดู! นี่เป็นการแปลง turtle ให้เป็น N-triples'. The 'Output' section shows the resulting N-Triples:

Number of triples parsed: 3

```
<http://dbpedia.org/resource/Pluto> dbo:hasSatellite <http://dbpedia.org/resource/Charon> .  
<http://dbpedia.org/resource/Charon> dbo:name <http://dbpedia.org/resource/Nyx> .  
<http://dbpedia.org/resource/Charon> dbo:hasSatellite <http://dbpedia.org/resource/Hybele> .
```

Semantic Annotation in the Web

RDFa, HTML Microdata, JSON-LD



Semantic Annotation in the Web

- ▣ เทคโนโลยีเว็บที่ใช้เอกสาร HTML ธรรมดा จะพบว่ามีปัญหาในการสืบค้นข้อมูลโดย search engine ทำให้ได้ข้อมูลที่ไม่ถูกต้องตรงตามความต้องการ หรือผู้ใช้ค้นหาสิ่งที่เขาต้องการไม่เจอ เว็บของเราไม่เป็นที่รู้จักของ Search Engine
- ▣ ปัจจุบันได้มีการพัฒนาต่างๆ ที่จะทำให้เอกสาร HTML มีส่วนที่อธิบายโครงสร้างข้อมูลที่ดีขึ้น ทำให้ search engine สามารถสืบค้นข้อมูลได้ง่ายขึ้น เข้าใจความหมายของข้อมูลมากขึ้น

38



Semantic Annotation in the Web

▣ วัตถุประสงค์ของหัวข้อนี้คือการใช้กลุ่มคำศัพท์ของ Standard Vocabularies เช่น schema.org, foaf, Dublin core, open vocab org เป็นต้น เข้ามาช่วยในการอธิบายข้อมูลในเอกสาร HTML

▣ ประโยชน์ต่อ

1. Search Engine สามารถค้นหาเว็บเพจที่มีการใส่คำอธิบายด้วย Standard Vocabularies ได้อย่างง่ายดาย
2. บุคคลอื่นจะค้นพบหน้าเว็บของเราได้ง่าย

39



Semantic Annotation in the Web

▣ จะมีหลายวิธีในการที่จะแทรกข้อมูลที่มีโครงสร้าง (structured data) ร่วมกับ Standard Vocabularies ภายในเอกสาร HTML

1. Domain-specific microformats (μformat) (ลักษณะแล้ว)
2. Generic RDFa
3. HTML5 Microdata (including schema.org)
4. JSON-LD

40

RDFa –RDF in HTML attributes



RDFa –RDF in HTML attributes

▣ RDFa เป็นภาษาที่อยู่ในกลุ่ม Serialization languages ภาษาหนึ่งที่ทำให้เราสามารถแทรกส่วนที่เป็น RDF model เข้าไปในเอกสาร (X)HTML ได้โดยการยังคงมีการใช้งาน (X)HTML attributes เดิมอยู่ แต่จะมีการแทรก new attributes เข้าไปได้ด้วย

▣ **RDFa 1.0** ถูกออกแบบบนพื้นฐานของเอกสาร XHTML
(W3C Recommendation 2008)

▣ **RDFa 1.1** ถูกปรับปรุงจาก 1.0 และถูกออกแบบบนพื้นฐานของ HTML5
(W3C Recommandataion June 2012) ประกอบด้วย

- **RDFa Lite1.1** (จะแนะนำให้รู้จักตัวพื้น)
- **RDFa 1.1**

41



RDFa –RDF in HTML attributes

▣ **RDFa** จะยังคงมีการใช้งาน (X)HTML attributes เดิมอยู่ เช่น href, src, span เป็นต้น และจะมีการเพิ่มเติม HTML attributes ใหม่ดังต่อไปนี้เข้าไปใน (X)HTML ด้วย

- vocab,
- typeof,
- property,
- resource,
- prefix

43



RDFa Lite 1.1

▣ ตอย. เช่น จากเอกสาร HTML ต่อไปนี้

```
<p>
  My name is Robert Lee and you can give me
  a phone via 287399388.
</p>
```

▣ เอกสารนี้จะเป็นเร็วที่แนะนำตัวเอง ถ้าเราต้องการให้มีการแทรกโครงสร้างของ **RDFa** เข้าไปเพื่อให้ search engine ค้นหาเอกสารของเรานี้ง่ายขึ้นจะทำอย่างไร

44

RDFa Lite 1.1

- เริ่มต้นจะต้องมีการกำหนดกลุ่มคำศัพท์ที่เราต้องการใช้ชื่อนามก่อน ซึ่งเราจะใช้กลุ่มคำศัพท์จาก FOAF ก็ให้กำหนด namespace ของ FOAF นี้อยู่ใน attribute "vocab"

```
<p vocab="http://xmlns.com/foaf/0.1/">
  My name is Robert Lee and you can give me
  a phone via 287399388.
</p>
```

- The FOAF (Friend Of A Friend) มีวัตถุประสงค์เพื่อกำหนดคำศัพท์ที่จะใช้ในหน้าเว็บเพจของเรา เช่น people, groups, companies, หรืออื่นๆ.
- สามารถเข้าไปที่ <http://xmlns.com/foaf/0.1/> เพื่อดูข้อมูลเพิ่มเติมได้

45

RDFa Lite 1.1

- ดูของคำศัพท์ที่ถูกสร้างอยู่ใน FOAF (<http://xmlns.com/foaf/0.1/>)

FOAF Overview

FOAF terms, grouped in broad categories.

FOAF Basics	Personal Info	Online Accounts / IM
<ul style="list-style-type: none"> Agent about name url title version mbox mbox_sha1sum primaryTopic description (object) family_name given_name firstName 	<ul style="list-style-type: none"> weblog blog interest currentProject pastProject place based_near workplace workForOrganization schoolHomepage friend publications interests notes memberships firstCheckin 	<ul style="list-style-type: none"> OnlineAccount EmailAccount OnlineCommerceAccount OnlineCarnivalAccount Interest accountServiceNamespace icqChatID imailChatID aimChatID msnChatID yahooChatID
Projects and Groups	Documents and Images	
<ul style="list-style-type: none"> Project Organization Group Member membershipClass isDefinedBy home 	<ul style="list-style-type: none"> Document Image Personnel-relatedDocument Image (page) primaryTopic byline shai version (format) thumbmall logo 	

RDFa Lite 1.1

- หลังจากนั้นก็จะต้องกำหนดประเภทของสิ่งที่เรากำลังพูดถึง (type of thing) ผ่านทาง attribute "typeof"

```
<p vocab="http://xmlns.com/foaf/0.1/"
  typeof="Person">
  My name is Robert Lee and you can give me
  a phone via 287399388.
</p>
```

- "typeof" จะเป็น attribute ของ RDFa, แต่ "Person" จะเป็นคำศัพท์ที่มีอยู่ใน foaf ที่มีลักษณะเป็น Class

47

RDFa Lite 1.1

- หลังจากนั้นเราก็สามารถกำหนด Properties ทั้งหมดของสิ่งที่เราอย่างเช่นว่าได้เพิ่มเติมแล้ว โดยใช้ attribute "property" ในการอธิบายว่าสิ่งที่เราพูดถึงนั้นคืออะไร

```
<p vocab="http://xmlns.com/foaf/0.1/" typeof="Person">
  My name is
  <span property="name">Robert Lee</span>
  and you can give me a phone via
  <span property="phone"> 287399388 .</span>
  
</p>
```

- "typeof" และ "property" คือ attributes ของ RDFa, แต่ "name", "phone" และ "img" จะเป็นคำศัพท์ที่มีอยู่ใน foaf

48

RDFa Lite 1.1

- เรายังสามารถแทรก attribute "resource" เพื่อเก็บสิ่งที่เรียกว่า identifiers ที่จะกำหนดให้เป็นประธาน (subject) ของ Properties ต่างๆ

```
<p vocab="http://xmlns.com/foaf/0.1/"
  resource="Robert" typeof="Person">
  My name is
  <span property="name">Robert Lee</span>
  and you can give me a phone via
  <span property="phone"> 287399388 .</span>
  
</p>
```

49

RDFa Lite 1.1

You can visit this Web Site:
<https://vocab.org/open/>

- ถ้าคำศัพท์ใน FOAF ไม่เพียงพอที่จะใช้อธิบาย Properties ทั้งหมดได้ เราสามารถกำหนดกลุ่มคำศัพท์เพิ่มเติมได้โดยใช้ Prefix

```
<p vocab="http://xmlns.com/foaf/0.1/"
  prefix="pa: http://open.vocab.org/terms/"
  resource="Robert" typeof="Person">
  My name is
  <span property="name">Robert Lee</span>
  and you can give me a phone via
  <span property="phone"> 287399388 .</span>
  
  <span property="pa:preferredAnimal">cat and dog</span>
</p>
```

50

HTML5 Microdata

μData



HTML5 Microdata (μData)

□ วัตถุประสงค์

- เพื่อช่วยให้เราสามารถใส่คำอธิบายเกี่ยวกับข้อมูลในเอกสาร HTML แบบมีโครงสร้างได้ง่ายขึ้น
- เพื่อให้ search engine สามารถเข้าดันหาข้อมูลและเก็บลงฐานข้อมูลของ search engine ได้ง่าย และจะช่วยให้คนอื่นสามารถสืบค้นข้อมูลจากเอกสาร HTML ของเราได้ง่าย เนื่องจากจะเป็นทรัพย์สินของคนอื่นได้ง่ายขึ้น
- เรายังสามารถเขียน Script เพื่อดึงข้อมูลที่เราต้องการออกนำไปใช้งานได้ง่ายโดยผ่านทาง DOM API นั่นเอง



HTML5 Microdata (μData)

- **HTML5 Microdata** จะช่วยให้เราสามารถใส่คำอธิบายเชิงความหมายของข้อมูลเข้าไปในเว็บเพจของเราได้
- เราสามารถใช้ Properties ของ Microdata แทรกเป็น attributes เข้าไปในเอกสาร HTML และกำหนดคำอธิบายที่นำมาจาก Schema.org เก็บเป็นค่าของ attributes เหล่านั้นได้

5



HTML5 Microdata (μData)

□ HTML5 Microdata specification จะประกอบด้วย

- Microdata vocabularies** จะเป็นกลุ่มคำศัพท์ที่นำมาจาก <http://schema.org>
- Microdata Global Attributes** ที่จะต้องแทรกอยู่ในเอกสาร html ได้แก่ attributes ต่อไปนี้
 - itemscope (ถูกใช้เพื่อบอกว่าตัวนี้คือ Microdata)
 - itemtype (เป็นการประกาศประเภท (type) ของสิ่งที่เราจะอธิบาย โดยนำคำมาจาก schema.org)
 - itemid (ใช้เพื่อบรรจุถึงชื่อ resource ที่จะอธิบาย)
 - itemprop (ใช้เพื่ออธิบายถึง property ของ resource)
 - itemref (ใช้เพื่อการอ้างอิงไปยัง resource อื่น)

6



HTML5 Microdata (μData)

□ นี่เป็นตัวอย่างการเขียนเอกสาร HTML ที่มีการผัง Microdata อยู่ภายใน

```
<p itemscope itemid="http://mydomain.com/Robert"
    itemtype="http://schema.org/Person">
  My name is
  <span itemprop="name">Robert Lee</span>
  and you can give me a phone via
  <span itemprop="telephone">287399388.</span>
  
  <span itemprop="jobTitle">Computer Support</span>
</p>
```

itemscope, itemid, itemtype, itemprop เป็นกลุ่มคำของ Microdata แต่ Person, name, telephone, image, และ jobTitle เป็นกลุ่มคำของ Schema.org

7



How to embed Microdata to HTML?

ตัวอย่างการนำ Microdata ไปใช้ในเอกสาร HTML ดังนี้

Schema.org/Product Examples

Example 1: [No Markup](#) Microdata RDFa JSON-LD Structure Example notes or example HTML without markup.

Kenmore White 17" Microwave

Rated 3.5/5 based on 11 customer reviews
\$55.00
In stock

Product description:
0.7 cubic feet countertop microwave. Has six preset cooking categories and convenience features like Add-A-Minute and Child Lock.

Customer reviews:

Not a happy camper - by Ellie, April 1, 2011
1/5 stars
The lamp burned out and now I have to replace it.

Value purchase - by Lucas, March 25, 2011
4/5 stars
Great microwave for the price. It is small and fits in my apartment.
...



Example of terms from Schema.org vocabulary

Schema.org Docs Schemas Validate About

Product
A Schema.org Type
Thing > Product
Any offered product or service. For example:

Property	Expected Value
Properties from Product	PropertyValue
additionalProperty	PropertyValue
aggregateRating	Text or URL
asIn	Audience

Note: Publishers should be aware that applications designed to use specific schema.org properties (e.g. <https://schema.org/width>, <https://schema.org/color>, <https://schema.org/gtin13...>) will typically expect such data to be provided using those properties, rather than using the generic property/value mechanism. The overall rating, based on a collection of reviews or ratings, of the item. An Amazon Standard Identification Number (ASIN) is a 10-character alphanumeric unique identifier assigned by Amazon.com and its partners for product identification within the Amazon organization (summary from Wikipedia's article).

Note also that this is a definition for how to include ASINs in Schema.org data, and not a definition of ASINs. In general, documents from Amazon for authoritative data on ASINs are most commonly encoded as text strings, but the [asin] property supports URL/URI as potential values too.

An intended audience, i.e. a group for whom something was created. Supersedes

Example of terms from Schema.org vocabulary

Schema.org Docs Schemas Validate About

AggregateRating
A Schema.org type
Thing > Intangible > Rating > AggregateRating
The average rating based on multi

Property	Expected Type	Description
Properties from AggregateRating		
itemReviewed	Thing	The item that is being reviewed/rated.
ratingCount	Integer	The count of total number of ratings.
reviewCount	Integer	The count of total number of reviews.
Properties from Rating		
author	Organization or Person	The author of this content or rating. Please note that author is special in that HTML 5 provides a special mechanism for indicating authorship via the rel tag. That is equivalent to this and may be used interchangeably.
bestRating	Number or Text	The highest value allowed in this rating system. If bestRating is omitted, 5 is assumed.
ratingExplanation	Text	A short explanation (e.g. one to two sentences) providing background context and other information that is helpful in understanding the rating. This is particularly applicable to ratings associated with "fact check" markup using ClaimReview.
ratingValue	Number	The rating for the content.
reviewAspect	Text	Usage guidelines:
worstRating	Number or Text	<ul style="list-style-type: none"> Use values from 0123456789 (Unicode 'DIGIT ZERO' (U+0030) to 'DIGIT NINE' (U+0039)) rather than superscripted Arabic numerals. Use a full stop (U+002E) rather than ';' to indicate a decimal point. Avoid using these symbols as a readability separator.

10

This Review or Rating is relevant to this part or facet of the itemReviewed.

This rating is relevant to the itemReviewed.

The rating for the content.

10

The rating for the content.



Example of using Microdata in HTML

schema.org/Product

Example 1

Microdata

Example encoded as Microdata embedded in HTML

```
<div itemscope itemtype="https://schema.org/Product">
  <span itemprop="name">Kenmore White 17" Microwave</span>
  
  <div itemprop="aggregateRating">
    <span itemprop="ratingValue">23.5</span></div>
    <div>
      <span>Rated based on <span>customer reviews</span></span>
    </div>
    <div itemprop="offers" itemscope itemtype="https://schema.org/Offer">
      <!-- price is 1000, a number, with locale-specific thousands separator and decimal mark, and the $ character is marked up with the machine-readable code "USD"-->
      <span>$1000.00</span><span>1,000.00</span>
      <span itemprop="priceCurrency">USD</span>
      <span itemprop="price">1000.00</span>
      <link itemprop="availability" href="https://schema.org/InStock" />In stock
    </div>
  <div>
    <span>Description</span>
    <span>0.7 cubic feet countertop microwave. Has six preset cooking categories and convenience features like Add-A-Minute and Child Lock.</span>
  </div>

```

Product description on

0.7 cubic feet countertop microwave. Has six preset cooking categories and convenience features like Add-A-Minute and Child Lock.

Microdata: Structure Data Testing Tools

- Google จะมีเครื่องมือที่เรียกว่า **Structure data testing tools** เพื่อช่วยในการทดสอบว่าเอกสาร HTML ของเรามีการฝังส่วนที่ใช้อธิบายข้อมูลที่เป็นโครงสร้างนี้เข้าไปด้วยหรือไม่
- เนื่องจาก google มีความจำเป็นที่จะต้องอ่านและใช้ข้อมูลที่มีโครงสร้างนี้เพื่อใช้ในการสืบค้นให้ได้อย่างถูกต้อง
- ตัวอย่างของ structure data testing tool ที่สร้างโดย google:
<https://validator.schema.org/>



Microdata: Structure Data Testing Tools

validator.schema.org

Schema.org Documentation Schemas About

คลิกที่ icon นี้เพื่อเลือกภาษาที่ต้องการ

ภาษาไทย

ภาษาอังกฤษ

ภาษาจีน

ภาษาฝรั่งเศส

ภาษาโปรตุเกส (โปรตุเกส)

ภาษาเวียดนาม

ภาษาญี่ปุ่น

ภาษาโปแลนด์

ภาษาเดนมาร์ก

ภาษาโรมาเนีย

Microdata: Structure Data Testing Tools

validator.schema.org

Schema.org Documentation Schemas About

ทดสอบ Structured Data ของคุณ

ลิงก์ของ URL

เข้าไปที่ผลลัพธ์

เราสามารถป้อน URL ของเอกสาร HTML ที่มี Microdata ที่ฝังอยู่ที่นี่แล้วกด Run Test



Microdata: Structure Data Testing Tools

หรือสามารถเลือกเมนูข้อมูลโค้ด เพื่อ copy และ parse ข้อมูลเอกสาร HTML ลงใน AreaText แล้วกด Run Test

กด Run Test

ถ้าโครงสร้างของเอกสารถูกต้อง ก็จะแสดงข้อผิดพลาดเป็น 0

และจะมีแสดงส่วนของ Microdata ที่ถูกอ่านได้จาก HTML มาแสดงให้เราดู



Microdata: Structure Data Testing Tools

ถ้าเราไม่ใส่ชื่ออยู่ใน "itemprop" ที่ไม่สอดคล้องกับ properties ต่างๆที่เก็บอยู่ใน "itemprop" ก็จะมีการแจ้งเตือนและออก警号ให้ทราบเพื่อท่องเที่ยวว่ามี properties ตัวไหนบ้างที่ไม่สอดคล้องกับ itemtype

คลิก

ถ้าไม่เข้าใจคิดพิจารณาโดยใช้ google สามารถดึงส่วนที่เป็นข้อมูลโครงสร้างที่ฝังอยู่ใน HTML ออกมาได้แล้วว่า google สามารถเข้าใจข้อมูลโครงสร้างที่แน่นหนาและสามารถงานนี้ไปใช้ในการสืบค้นทันทีได้

JSON-LD



JSON-LD

- เพื่อจำกัดความต้องการที่จะพัฒนา RDF ให้อยู่ในรูปแบบของ JSON เพื่อให้ hakka พัฒนาโปรแกรมสามารถนำไปใช้งานและจัดการกับ RDF data ได้ง่าย
- **JSON-LD (JSON-Linked Data)** จึงได้ถูกนำเสนอขึ้นมาซึ่งจะเป็น lightweight Linked Data format ที่มีความง่ายที่อ่านและเขียน



JSON-LD

- ❑ JSON-LD จะมีโครงสร้างเหมือน JSON และสอดคล้องกับ RDF graphs
- ❑ JSON-LD จะสามารถถูกแปลงจาก JSON documents ให้เป็น RDF ได้ง่าย โดยมีการเปลี่ยนแปลงที่น้อยมาก
- ❑ JSON-LD จะถือว่าเป็นรูปแบบข้อมูลที่น่าสนใจเพื่อการพัฒนาโปรแกรมต่างๆ ได้ง่าย รวมถึงการใช้งานร่วมกับ REST Web services และ unstructured databases เช่น Apache CouchDB and MongoDB

24



JSON-LD

Developer

JSON-LD is available in a number of popular programming environments. Each implementation of JSON-LD listed below is fully conforming to the official JSON-LD specifications.

Javascript	Python	Ruby	Go
jsonld.js <small>1.1</small> jsonld-streaming-parser.js <small>1.1</small> jsonld-streaming-serializer.js <small>1.1</small> rdf-parse.js <small>1.1</small>	PyLD <small>1.9</small> RDFLib <small>1.9</small>	JSON-LD for RDF.rb <small>1.1</small>	JSON-goLD <small>1.1</small>
Java	C#	Erlang / Elixir	PHP
Titanium <small>1.1</small> JSONLD-JAVA <small>1.0</small>	dnnNetRDF <small>1.1</small> json-ld.net <small>1.0</small>	jsonld-ex <small>1.0</small>	php-json-ld <small>1.0</small> JsonLD <small>1.0</small>
Rust	TypeScript		
json-ld <small>1.1 (WIP)</small>	jsonld-lint <small>1.1 (WIP)</small>		

JSON-LD สามารถใช้งานกับภาษาโปรแกรมต่างๆ ที่ได้รับความนิยมจากผู้พัฒนาโปรแกรมเหล่านี้

25



JSON-LD

Example of JSON document

```
{
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Key: value pairs

ตัวอย่างนี้เป็นตัวอย่างเอกสาร JSON ที่เราคุ้นเคยกัน ที่ประกอบด้วย key value pair ที่มีโครงสร้างง่ายๆ การเขียนโครงสร้าง JSON แบบนี้จะทำให้การใช้งานมีความถูกต้อง ทั้งคนและ machine จะไม่มีความซ้ำซ้อน name จะไม่ซ้ำกับ homepage และ image ที่เขียนนั้น และถ้ามีเอกสารหลักของเอกสาร ก็จะสามารถใช้อ้างอิงได้ เช่น identifier ที่อ้างอิงไปยัง namespace ของ schema.org ที่เป็น key เหล่านี้เพื่อทำให้ unique ไม่มีความถูกต้อง หรือเกิดขึ้น การ reuse และ sharing เอกสารก็จะทำได้ง่าย

26



JSON-LD

Example of JSON-LD document

```
{
  "https://schema.org/name": "Manu Sporny",
  "https://schema.org/url": "http://manu.sporny.org/",
  "https://schema.org/image": "http://manu.sporny.org/images/manu.png"
}
```

ตัวอย่างนี้เป็นการใช้คำศัพท์จาก schema.org เพื่อแทรกเข้าไปในเอกสาร JSON นี้ เพื่อ แก้ไขปัญหาความถูกต้องของคำและทำให้ key อยู่รูปแบบของ URI

27



JSON-LD

Try to use
<https://json-ld.org/playground/>

Example of JSON-LD document

Try to use <https://json-ld.org/playground/>

JSON-LD Playground

Play around with JSON-LD markup by typing out some JSON below and seeing what gets generated from it at the bottom of the page. Pick any of the examples below to get started.

NOTES:

- The playground uses jsonld.js which conforms to JSON-LD 1.1 syntax (errata), API (errata), and framing (errata).
- Other related playgrounds: Classic JSON-LD 1.0 Playground | RDF Distiller | Verifiable Credentials Playground

Examples: Person Event Place Product Recipe Library Activity

JSON-LD Input

```
{"@context": "https://schema.org", "name": "Manu Sporny", "url": "http://manu.sporny.org/", "image": "http://manu.sporny.org/images/manu.png"}
```

28



JSON-LD

Try to use
<https://json-ld.org/playground/>

Example of JSON-LD document



29



JSON-LD

Another Example of JSON-LD document

```
{
  "@context": "https://schema.org",
  "name": "Manu Sporny",
  "url": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

- อย่างไรก็ตาม เราสามารถเขียนเครื่องรับแบบหนึ่งเดียว มีการประกาศ "@context" เพื่อเก็บ standard vocabulary ที่เราจึงใช้อ้างอิง ซึ่งได้แก่ เว็บของ schema.org หลังจากนั้นก็เรียกใช้ ค่าตัวพัทที่อยู่ใน schema.org ได้เลย โดยไม่ต้องปะ prefix ข้างหน้า คือทำให้อยู่ในรูปแบบของ key: value pairs เมื่ออนดิม



JSON-LD

Another Example of JSON-LD document

```
{
  "@context": "https://schema.org",
  "@id": "https://www.mydomain.com/P0001",
  "@type": "Product",
  "name": "T-Shirt",
  "image": "http://www.mydomain.com/images/P0001.png",
  "manufacturer": {
    "@id": "https://www.product.com/C0001",
    "@type": "https://www.product.com/Company",
    "name": "Xuzang Company"
  }
}
```

31



Example of linking JSON-LD in HTML

The screenshot shows a browser window with the URL [schema.org/Person](#). The page content includes a JSON-LD script block:

```

<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "Person",
  "name": "Jane Doe",
  "address": {
    "@type": "PostalAddress",
    "addressLocality": "Seattle",
    "addressRegion": "WA",
    "postalCode": "98052",
    "streetAddress": "20341 Whitworth Institute"
  },
  "colleague": [
    "http://www.xyz.edu/students/alicejones.htm",
    "http://www.xyz.edu/students/bobsmith.html"
  ],
  "email": "mailto:jane-doe@xyz.edu",
  "image": "janedoe.jpg",
  "jobTitle": "Professor",
  "name": "Jane Doe",
  "telephone": "(425) 123-4567",
  "url": "http://www.janedoe.com"
}</script>
```

Annotations highlight specific parts of the JSON-LD:

- A red box surrounds the entire JSON-LD block, labeled "ตัวเรามีการกำหนด @id หลักให้เอกสาร แต่ไประบุ @type เป็น 'Person' เลย กรณีนี้คอมพิวเตอร์จะทำการสร้าง Blank node ขึ้นมาให้ท่านที่เป็น resource หลัก (Subject) ที่จะถูกเก็บอยู่ใน @id และกำหนดให้ Blank node นั้นมี type เป็น 'Person' อีกที"
- A red box highlights the "address" property, labeled "ด้วยนี่ก็เช่นกัน ไม่มีการกำหนด @id ให้เป็น value ของ property 'address' และมีการระบุ @type อย่างเดียว คอมพิวเตอร์จะทำการสร้าง Blank node ขึ้นมาให้ท่านที่เป็น resource ที่เก็บอยู่ใน @id และกำหนดให้ Blank node นั้นมี type เป็น 'PostalAddress'"
- A red box highlights the "image" property, labeled "เอกสาร JSON-LD นี้จะสามารถนำไปแทรกอยู่ในเอกสาร HTML ได้โดยตรง เพื่อช่วยให้ google สามารถสืบค้นข้อมูลในเอกสาร HTML ได้ง่ายและเร็วขึ้น"

RDF/RDF Schema Feature



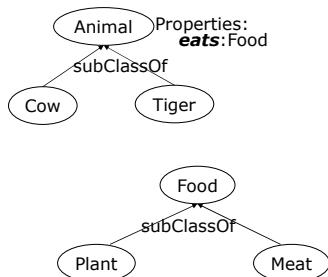
RDF/RDF Schema Features

- ❑ RDF/RDF Schema จะเกี่ยวกับการสร้างสิ่งต่อไปนี้:

- Classes and subclassOf relationship,
- Properties and subproperty relationship,
- domain and range restrictions, and
- instances of classes

4

RDF Schema is a kind of Taxonomies!



5



RDF Schema Expressed in Turtle

```
@base <http://www.animal.com/> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

```
<Cow>  
a rdfs:Class;  
rdfs:subClassOf <Animal>.  
<Tiger>  
a rdfs:Class;  
rdfs:subClassOf <Animal>.  
<Animal>  
a rdf:Class.  
<Plant>  
a rdfs:Class;  
rdfs:subClassOf <Food>.  
  
<Meat>  
a rdfs:Class;  
rdfs:subClassOf <Food>.  
<Food>  
a rdfs:Class.  
<eats>  
a rdf:Property;  
rdfs:domain <Animal>;  
rdfs:range <Food>.
```

6



Limitations of the Expressive Power of RDF Schema (1)

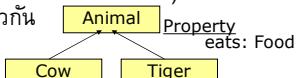
❑ Local scope of properties:

- rdfs:range จะใช้ในการกำหนดประเภทข้อมูลของ property (type of property's value) เช่น ถ้ามีการกำหนดให้ property "eats" มี range เป็น class Food

Ex. <eats>
a rdf:Property;
rdfs:domain <Animal>;
rdfs:range <Food>.

กรณีแบบที่จะไม่ได้ก่อตัว

- จากดูใน taxonomy จะพบว่า instances ของ class Cow (ซึ่งได้แก่ วัวและตัว) จะสามารถกิน (eats) ได้ทั้ง Plant และ Meat และ instance ของ class Tiger (ซึ่งได้แก่ เสือและเสือตัว) ก็จะสามารถกิน (eats) ได้ทั้ง Plant และ Meat เช่นเดียวกัน



Limitations of the Expressive Power of RDF Schema (1)

❑ Local scope of properties: (2)

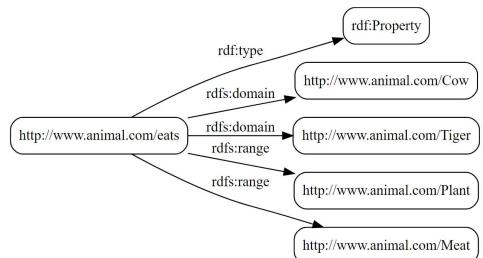
- ใน RDF Schema, ถ้าเราประกาศ domain และ range ของ property "eats" ให้เป็น classes ที่แตกต่างกัน ดังดอยต่อไปนี้

Ex. <eats>
a rdf:Property;
rdfs:domain <Cow>;
rdfs:range <Plant>.

<eats>
a rdf:Property;
rdfs:domain <Tiger>;
rdfs:range <Meat>.

- การเขียนดังดอยข้างต้นนี้ ไม่สามารถพูดได้ว่า Cow eats only plants, ส่วน Tiger eats only meat

Limitations of the Expressive Power of RDF Schema (1)



- จากด้วยข้างต้น จะหมายความว่า **eats** จะสามารถถูกใช้กับ Animals ทุกตัวที่เป็นทั้ง Cow และ Tiger (**instances** ที่มี type เป็นทั้ง Cow และ Tiger) ซึ่งจะสามารถ eats ได้ทั้ง Plant and Meat

9

Limitations of the Expressive Power of RDF Schema (2)

□ Disjointness of classes

- บางครั้งเราก็อยากที่จะสร้าง classes ที่มีลักษณะที่เป็น **disjoint** classes กัน เช่น class **Male** และ **Female** ต้องไม่มีสมาชิกที่ซ้ำกัน

➤ แต่ RDF Schema จะมีเฉพาะเรื่อง **subclass** relationships เท่านั้น เช่น class **Female** จะเป็น **subclass of Person** เป็นต้น

□ Boolean combinations of classes

- บางครั้งเราก็อยากสร้าง classes ที่เป็นการใช้คุณสมบัติ **union**, **intersection**, and **complement** กันระหว่าง classes เช่น

➤ class **Person** เกิดจากการ **disjoint union** กันของ classes **Male** และ **Female**

➤ RDF Schema จะไม่สามารถทำในเรื่องเหล่านี้ได้

10

Limitations of the Expressive Power of RDF Schema (3)

□ Cardinality restrictions

- RDFS ไม่สามารถจำกัดจำนวน property values ได้
- เช่น Person จะต้องมี **parents** ไม่เกิน 2 คน หรือวิชา (Course) จะสามารถถูกสอนโดยอาจารย์ (Lecturer) อย่างน้อย 1 คน

□ Special characteristics of properties

- RDFS จะไม่สนับสนุนคุณสมบัติ **Transitive**, หรือ **inverse property** (เช่น "eats" และ "is eaten by") หรืออื่นๆ

ดังนั้น เราจึงต้องการ **ontology language** ที่มีคุณสมบัติที่ดีกว่า **RDF Schema** !!

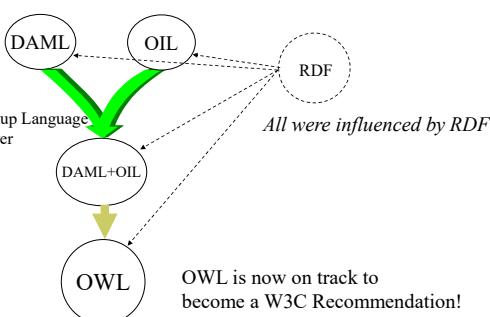
11

DAML/OIL

- August 2000: รัฐบาลอเมริกาได้ให้ทุนสนับสนุนโครงการเพื่อพัฒนา **DAML-ONT(DAPAR Agent Markup language)** บนพื้นฐานของ RDF class
- ในขณะเดียวกันทางฝั่งยุโรป ได้มีการพัฒนาภาษาที่เรียกว่า **OIL(Ontology Inference Layer)** บนพื้นฐานของ **Description Logic (DL)** และ **frame-based systems in AI**
- ต่อมาได้มีการประชุมกันเพื่อทำการรวม DAML กับ OIL เพื่อสร้างเป็นภาษาที่เรียกว่า **DAML+OIL**
- RDF จะถูกใช้เป็นพื้นฐานของ **DAML+OIL**
- ซึ่งต่อมาได้มีการพัฒนา **DAML+OIL** ลึกให้อยู่ในรูปแบบของ **OWL (Ontology Web Language)** ซึ่งเป็นภาษาที่องค์กร W3.org แนะนำ

12

Ontology Web Language



13

The Purpose of OWL

- วัตถุประสงค์ของ OWL นั้นเหมือนกับ RDF Schemas นั้นคือเพื่อกำหนด classes, properties และความสัมพันธ์ระหว่าง classes รวมไปถึงการเขียนต่อของ classes และ properties

➤ **RDF Schema** จะช่วยให้เราสามารถสร้างโครงสร้าง ontology พื้นฐานได้ แต่จะมีข้อจำกัดของ inference engine ในการอนุมานข้อมูลเพื่อทำความสัมพันธ์ใหม่ๆ ที่ซ่อนอยู่

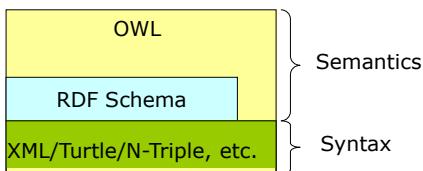
➤ แต่ **OWL** จะช่วยให้เราสามารถสร้างคุณสมบัติเพิ่มเติมเข้าไปใน ontology ทำให้ inference engine มีความสามารถในการอนุมานข้อมูลเพิ่มขึ้นได้ดีกว่า **RDF Schema** เพื่อหาความสัมพันธ์ใหม่ๆ ที่ซ่อนอยู่ใน ontology

OWL = RDF Schema + more

14



OWL- enables machine-processable semantics



Using OWL to Define Properties

15



Defining Property Characteristics

- RDF Schema จะช่วยให้เราสามารถอธิบายคุณสมบัติของ property ได้ 3 แบบ
 - **rdfs:domain:**
 - **rdfs:range:**
 - **rdfs:subPropertyOf:**
- Note: OWL documents ก็จะยังคงสามารถใช้ rdfs:range, rdfs:domain, และ rdfs:subPropertyOf ได้เช่นเดียวกันกับ RDF schema
- สำหรับ slides ในหน้าต่อไป จะแสดงคุณสมบัติที่มีเพิ่มเติมเข้าไปใน OWL ซึ่งจะไม่มีใน RDF schema
 - ซึ่งจะแสดงให้เห็นด้วยว่า คุณสมบัติที่เพิ่มเติมเข้าไปใน OWL นี้จะช่วยให้ inference engine สามารถอนุมานความสัมพันธ์อื่นๆเพิ่มเติมได้มากขึ้นอีก

18

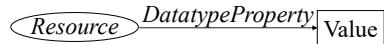


ObjectProperty vs. DatatypeProperty

- An **ObjectProperty** relates one **Resource** to another **Resource**:



- A **DatatypeProperty** relates a **Resource** to a **Literal** or an **XML Schema datatype**:



Defining Properties in OWL vs. RDF Schema

```
<taughtBy>
  a rdf:Property;
  rdfs:domain <Subject>;
  rdfs:range <AcademicStaff>.
<credit>
  a rdf:Property;
  rdfs:domain <Subject>;
  rdfs:range xsd:nonNegativeInteger.
```

RDFS

```
<taughtBy>
  a owl:ObjectProperty;
  rdfs:domain <Subject>;
  rdfs:range <AcademicStaff>.
<credit>
  a owl:DatatypeProperty;
  rdfs:domain <Subject>;
  rdfs:range xsd:nonNegativeInteger.
```

OWL



The OWL Namespace

```
@base <http://www.university.ac.th/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

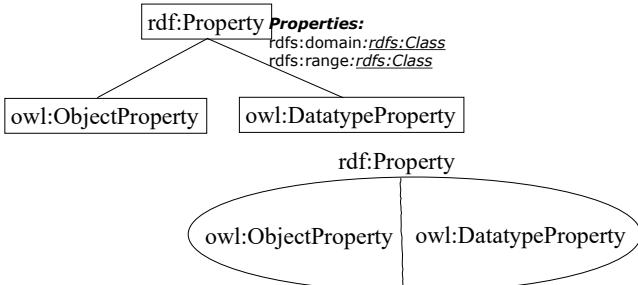
```
<taughtBy>
  a owl:ObjectProperty;
  rdfs:domain <Subject>;
  rdfs:range <AcademicStaff>.
<credit>
  a owl:DatatypeProperty;
  rdfs:domain <Subject>;
  rdfs:range xsd:nonNegativeInteger.
```

20

21



owl:ObjectProperty and owl:DatatypeProperty
are subclasses of rdf:Property



22



OWL defines additional properties from RDFS

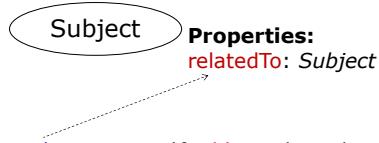
- A property in OWL may be defined to be:
 - A Symmetric property.
 - A Transitive property.
 - The Inverse of another property.
 - A Functional property.
 - An Inverse Functional property.

23



Symmetric Properties

- ถ้า property “relatedTo” มี domain และ range คือ class Subject และมีการกำหนดให้เป็น Symmetric Properties



A Symmetric property – if subject A has relatedTo subject B
then subject B has relatedTo subject A.

24



Syntax for indicating that a property is Symmetric

```

@base <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<relatedTo>
  a owl:ObjectProperty,
  owl:SymmetricProperty;
  rdfs:domain <Subject>;
  rdfs:range <Subject>.
  
```

This means that
`<relatedTo>`
`a owl:ObjectProperty;`
`owl:SymmetricProperty;`

อ่านว่า “relatedTo” มี type เป็น ObjectProperty และมี type
เป็น SymmetricProperty ด้วย

25



Symmetric Property (2)

- ถ้า relatedTo ถูกกำหนดในเอกสาร OWL ให้เป็น Symmetric property และถ้าเราอ่านเอกสารที่เป็น instance document ดังตัวอย่างนี้

```

@base <http://www.university.edu/subject/> .
@prefix univ: <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<322735>
  a univ:Subject;
  univ:relatedTo <322736>.
  
```

- ถ้า relatedTo ถูกกำหนดให้เป็น Symmetric property, inference engine จะสามารถอนุมานได้ว่า The subject 322735 relatedTo the subject 322736 และ The subject 322736 relatedTo the subject 322735.

26



Symmetric Property (3)

- จริงๆแล้วเรายังสามารถยุบเขียน OWL schema level และ OWL instance level ให้เป็น turtle file เพียง file เดียวได้ด้วย:

sym.ttl

```

@base <http://www.university.edu/> .
@prefix sub: <http://www.university.edu/subject/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<relatedTo>
  a owl:ObjectProperty,
  owl:SymmetricProperty;
  rdfs:domain <Subject>;
  rdfs:range <Subject>.
  
```

Schema Level

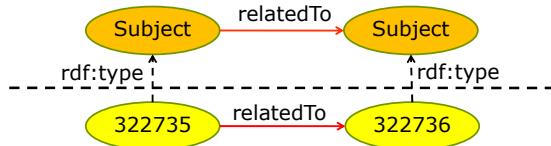
```

sub:322735
  <relatedTo> sub:322736.
  
```

Instance Level

27

Symmetric Property (4)

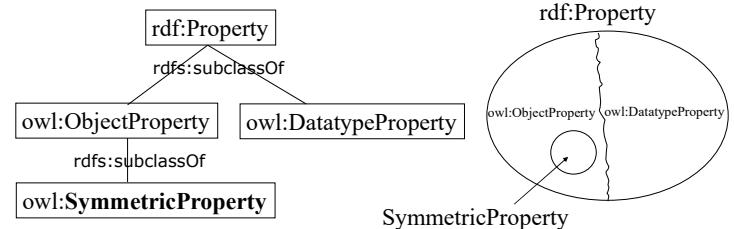


- จากข้อมูลข้างต้นเมื่อ load เข้าสู่ inference engine จะทำให้ inference engine สามารถอนุมานข้อมูลได้ว่า



28

owl:SymmetricProperty is a subclass of **owl:ObjectProperty**



SymmetricProperty

SymmetricProperty จะมี range เป็น Resource (classes หรือ instances) แต่จะไม่สามารถมี range เป็น Literal หรือ datatype อีกต่อไป

Transitive Properties

- ถ้า property “**hasPrerequisite**” (มีวิชาที่ต้องลงเรียนก่อนหน้า) มี domain และ range คือ class **Subject** และมีการกำหนดให้เป็น **Transitive Properties**



A Transitive property - if subject A has prerequisite to subject B, and subject B has prerequisite to subject C then subject A has prerequisite to subject C.

30

Syntax for indicating that a property is Transitive

```

@base <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<hasPrerequisite>
  a owl:ObjectProperty,
    owl:TransitiveProperty;
  rdfs:domain <Subject>;
  rdfs:range <Subject>.
  
```

Transitive Property (2)

- ถ้า **hasPrerequisite** ถูกกำหนดในเอกสาร OWL ให้เป็น **Transitive property** และถ้าเรามีเอกสารที่เป็น instance document ดังด้วยต่อไปนี้

```

@base <http://www.university.edu/subject/> .
@prefix univ: <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<322732>
  univ:hasPrerequisite <322731>.

<322731>
  univ:hasPrerequisite <322730>.
  
```

Since **hasPrerequisite** has been defined to be a **Transitive** property. If subject 322732 **hasPrerequisite** the subject 322731 and subject 322731 **hasPrerequisite** the subject 322730, we can infer that subject 322732 **hasPrerequisite** the subject 322730 too.

Transitive Property (3)

จริงๆแล้วเราสามารถยืนยัน OWL schema level และ OWL instance level ให้เป็น turtle file เพียง file เดียวได้ด้วย:

```

@base <http://www.university.edu/> .
@prefix sub: <http://www.university.edu/subject/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<hasPrerequisite>
  a owl:ObjectProperty,
    owl:TransitiveProperty;
  rdfs:domain <Subject>;
  rdfs:range <Subject>.
  
```

sym.ttl

```

sub:322732
  <hasPrerequisite> sub:322731.

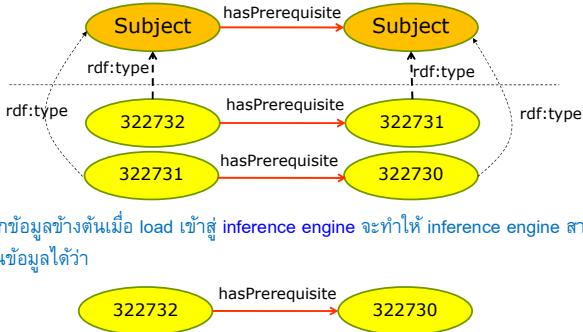
sub:322731
  <hasPrerequisite> sub:322730.
  
```

Schema Level

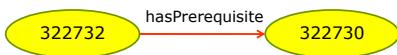
Instance Level

33

Transitive Property (4)

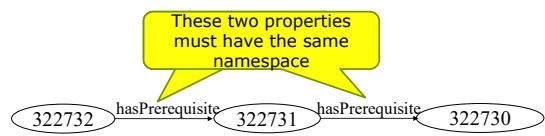


- จากข้อมูลข้างต้นเมื่อ load เข้าสู่ inference engine จะทำให้ inference engine สามารถอนุมานข้อมูลได้ว่า

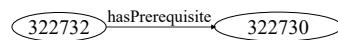


34

Transitive Property (5)

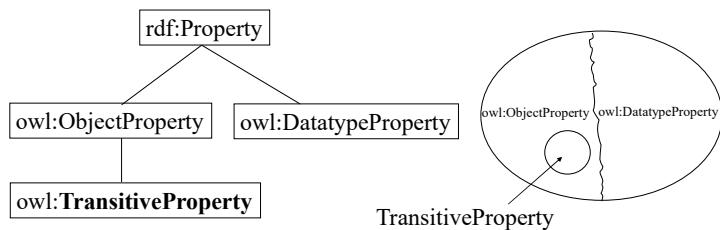


- ถ้า `hasPrerequisite` ถูกกำหนดให้เป็น Transitive Property ดังนั้น Inference engine จะสามารถอนุมานข้อมูลได้ว่า



35

`owl:TransitiveProperty` is a subclass of `owl:ObjectProperty`



`TransitiveProperty` จะมี range เป็น Resource (classes หรือ instances) แต่จะไม่สามารถมี range เป็น Literal หรือ datatype อีกที่ได้

Syntax for indicating that a property is the inverse of another property

```
@base <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<teaches>
  a owl:ObjectProperty;
  rdfs:domain <AcademicStaff>;
  rdfs:range <Subject>.

<taughtBy>
  a owl:ObjectProperty;
  owl:inverseOf <teaches>.
```

38

Inverse Properties (2)

□ ถ้ามีการสร้าง instance document ดังต่อไปนี้

```
@base <http://www.university.edu/staff/> .
@prefix univ: <http://www.university.edu/> .
@prefix sub: <http://www.university.edu/subject/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<ST949318>
  univ:teaches sub:CS731.
```

จากข้อมูลใน instance document ข้างบนจะได้ว่า
(**ST949318, teaches, CS731**)

ถ้ามีการกำหนดให้ `teaches` และ `taughtBy` เป็น Inverse properties กันดังนั้น inference engine จะสามารถอนุมานข้อมูลได้ว่า: (**CS731, taughtBy, ST949318**)

37



Inverse Properties (3)

➤ จริงๆแล้วเราสามารถยุบเขียน OWL schema level และ OWL instance level ให้เป็น turtle file เพียง file เดียวได้ด้วย:

```
@base <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix stf: <http://www.university.edu/staff/> .
@prefix sub: <http://www.university.edu/subject/> .
```

```
<teaches>
  a owl:ObjectProperty;
  rdfs:domain <AcademicStaff>;
  rdfs:range <Subject>.
```

Schema Level

```
<taughtBy>
  a owl:ObjectProperty;
  owl:inverseOf <teaches>.
```

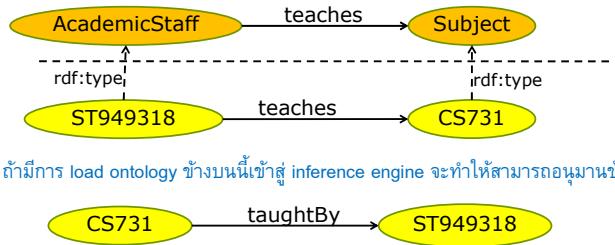
Instance Level

```
stf:ST949318
<teaches> sub:CS731.
```

40

Inverse Properties (4)

□ ถ้า taughtBy ถูกกำหนดให้เป็น inverse property ของ teaches property



41



Summary of the different ways to characterize properties

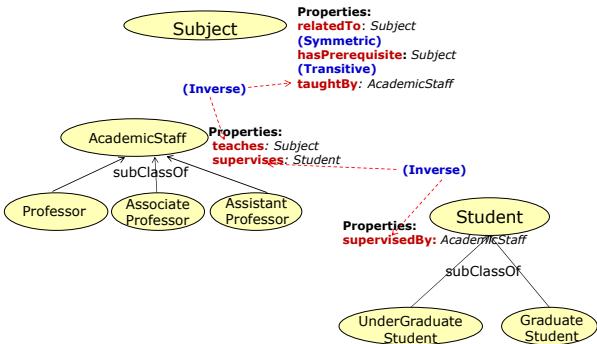
□ สรุปใน slides ก่อนหน้านี้จะมีการอธิบายถึงคุณสมบัติของ Properties ต่างๆดังต่อไปนี้:

- A Symmetric property.
- A Transitive property.
- The Inverse of another property.
- A Functional property.
- An Inverse Functional property.

42



Summary of Properties for the University Taxonomy



43



Inferences we can make now that we have characterized the properties

```
@base <http://www.university.edu/subject/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix univ: <http://www.university.edu/> .

<322731>
  a univ:Subject;
  univ:taughtBy stf:DavidLee;
  univ:relatedTo <322720>.
```

What can we infer about these resources?

1. DavidLee 322731
2. 322720 322731
3. DavidLee is an
4. 322720 is a

44



Inferences we can make now that we have characterized the properties

```
@base <http://www.university.edu/subject/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix univ: <http://www.university.edu/> .

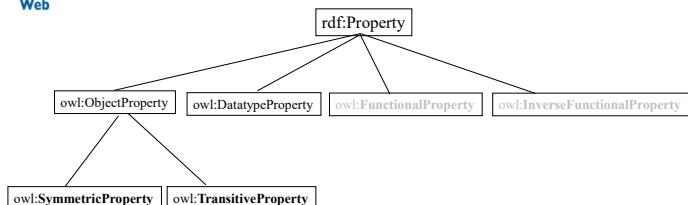
<322731>
  a univ:Subject;
  univ:taughtBy stf:DavidLee;
  univ:relatedTo <322720>.
```

What we can infer about:

1. DavidLee teaches 322731. (Since teaches is the inverse of taughtBy)
2. 322720 relatedTo 322731. (Since relatedTo is symmetric)
3. DavidLee is an AcademicStaff. (Since the range of taughtBy is an AcademicStaff)
4. 322720 is a Subject. (Since the range of relatedTo is a Subject)

45

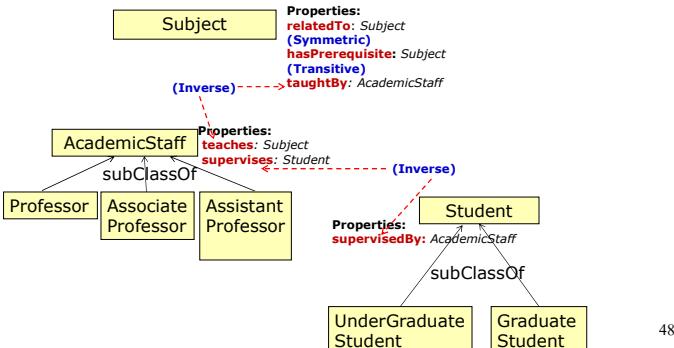
Hierarchy of the property classes



Consequences:

1. **SymmetricProperty** and **TransitiveProperty** can only be used to relate Resources to Resources.
2. FunctionalProperty and InverseFunctionalProperty can be used to relate Resources to Resources, or Resources to an RDF Schema Literal or an XML Schema datatype.

LAB: use Protégé to convert this taxonomy into turtle code



48



Defining Classes in OWL

- OWL classes จะช่วยให้เราสามารถเพิ่มเติมคุณสมบัติต่างๆ ใน ontology ได้มากกว่า RDF Schema

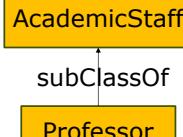
▪ การสร้าง OWL class จะทำได้ด้วยการใช้คำสั่ง `owl:Class`.

RDFS

```
<Professor>
a rdfs:Class;
rdfs:subClassOf <AcademicStaff> .
```

OWL

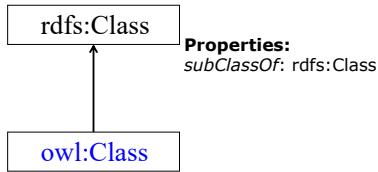
```
<Professor>
a owl:Class;
rdfs:subClassOf <AcademicStaff> .
```



4



owl:Class is a subclass of rdfs:Class



5

Using OWL Class to set Equivalent classes

`owl:equivalentClass`



Defining a class to be equivalent to another class

- `owl:equivalentClass` จะเป็นคุณสมบัติหนึ่งที่สามารถเพิ่มเข้าไปใน `owl:Class` ได้เพื่อใช้อธิบายถึงความเท่าเทียมกันของ classes

- ต.ย. เช่น ถ้าต้องการอธิบายว่า class `AdultFemaleHuman` มีความเท่าเทียมกันกับ class `Woman` ดังนี้



- จะหมายความว่า **สมาชิก (instances)** ของ class `AdultFemaleHuman` และของ class `Woman` จะถูกรวมเข้าด้วยกันเป็นสมาชิกกลุ่มเดียวกัน

7



Defining a class to be equivalent to another class

```

@prefix : <http://www.mydomain.com/person/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:AdultFemaleHuman
a owl:Class;
owl:equivalentClass :Woman.
:Woman
a owl:Class.
  
```

```

#define class
:AdultFemaleHuman
a owl:Class;
owl:equivalentClass :Woman.
:Woman
a owl:Class.

#define instance
:3168812422
a :AdultFemaleHuman.
:3287789999
a :Woman.
  
```

Use
Protégé
to create
this
ontology
too.

8

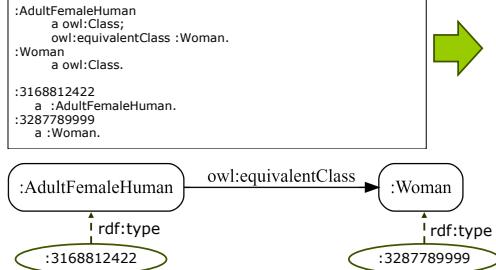
Defining a class to be equivalent to another class

```
@prefix : <http://www.mydomain.com/person/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:AdultFemaleHuman
a owl:Class;
owl:equivalentClass :Woman.

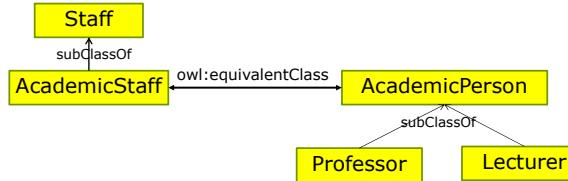
:Woman
a owl:Class.

:3168812422
a :AdultFemaleHuman.
:3287789999
a :Woman.
```



- ถ้ามีการถาม inference engine ให้แสดง instances ทั้งหมดของ `:Woman` ออกมานา (หรือแสดง instances ทั้งหมดของ `:AdultFemaleHuman` ออกมานา) ผลลัพธ์ที่ได้จะเป็นการแสดง instances ที่นำมาจากทั้งสองคลาส ซึ่งได้แก่ (`:3168812422` และ `:3287789999`)

Defining a class to be equivalent to another class



- ถ้ามีการถาม Inference engine ให้แสดง instances ทั้งหมดของ `:AcademicStaff` (หรือแสดง instances ทั้งหมดของ `:AcademicPerson`) ผลลัพธ์ที่ได้จะแสดง instances ทั้งหมดที่ดึงมาจากทั้งสองคลาส (ได้แก่ instances จาก `:AcademicStaff`, `:Professor` และ `:Lecturer`)

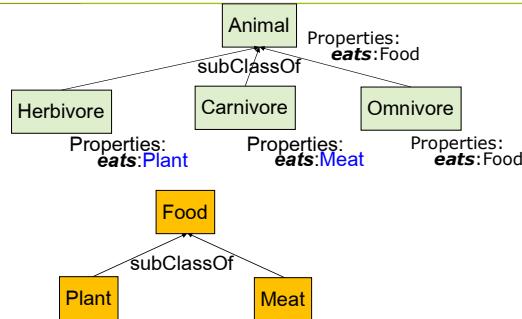
10

Using OWL Class to change the local scope of property

`owl:allValuesFrom`



Example: Sometimes a class needs to restrict the range of a property



ในการนี้ที่ต้องการจำกัด range ของ property "eats" คือ ถ้า "eats" ถูกใช้กับ class "Herbivore" จะต้องมี range เป็น class "Plant" แต่ถ้า "eats" ถูกใช้กับ class "Carnivore" จะต้องมี range เป็น class "Meat" เท่านั้น แต่ถ้า "eats" ถูกใช้กับ class "Omnivore" ก็จะมี range เป็น class Food

Restrict the range of a Property

- Property `eats` จะมี range เป็น class `Food` (ที่ประกอบด้วย `Meat` และ `Plant`) ซึ่งถ้าเป็น RDF schema จะมีการสืบทอด property `eats` ไปยัง class `Herbivore`(สัตว์กินพืช) และ `Carnivore` (สัตว์กินเนื้อ) ซึ่งจะทำให้ class `Herbivore` และ `Carnivore`สามารถ `eats` ได้ทั้ง `Plant` และ `Meat`
- แต่ถ้าเราต้องการเปลี่ยนแปลง range ของ property `eats` เมื่อมีการใช้กับ class ใด class หนึ่ง เช่น ถ้า `eats` ถูกใช้กับ class `Herbivore` ให้เปลี่ยน range จาก `Food` เป็น `Plant` แทน ถ้า `eats` ถูกใช้กับ class `Carnivore` ให้เปลี่ยน range จาก `Food` เป็น `Meat` แทน
- เราจะมีวิธีการเปลี่ยนแปลงค่า range ของ property นี้ได้อย่างไรด้วย OWL?

13



Defining eats (when used in Herbivore) to have allValuesFrom the Plant class

```
@base <http://www.animal.com/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ani: <http://www.animal.com/> .
```

Define the classes

```
<Animal> a owl:Class .
```

```
<Herbivore> a owl:Class ;
```

```
rdfs:subClassOf <Animal> ;
```

```
rdfs:subClassOf
```

```
[ a owl:Restriction ;
owl:onProperty <eats> ;
owl:allValuesFrom <Plant>
```

1.



Cow is a subclass of an "Anonymous class"

- Two forms of writing Herbivore is a subclass of an "Anonymous class"

```
<Herbivore> a owl:Class ;
rdfs:subClassOf <Animal> ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:onProperty <eats> ;
owl:allValuesFrom <Plant>
].
```

Anonymous Class

```
<Herbivore> a owl:Class ;
rdfs:subClassOf <Animal> ;
rdfs:subClassOf _:x1.
_:x1 a owl:Restriction ;
owl:onProperty <eats> ;
owl:allValuesFrom <Plant>.
```

Anonymous Class

15



Herbivore is a subclass of an "Anonymous class"

Anonymous Class

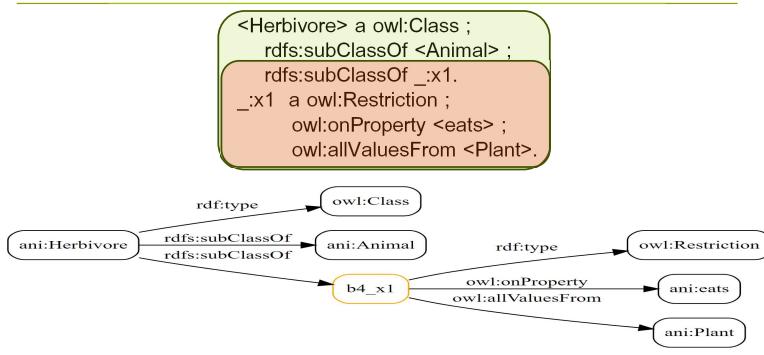
```
<Herbivore> a owl:Class ;
rdfs:subClassOf <Animal> ;
rdfs:subClassOf _:x1.
_:x1 a owl:Restriction ;
owl:onProperty <eats> ;
owl:allValuesFrom <Plant>.
```

อ่านว่า Class "Herbivore" เป็น subClass ของ class "Animal" และยังเป็น subClass ของอีก class ที่เป็น Anonymous class (class เปล่าๆไม่มีชื่อ หรืออาจตั้งชื่อว่า _:x1) ซึ่ง class "Herbivore" นี้จะมีการใช้ property "eats" ซึ่งมีค่าข้อมูลทุกค่า (all values) ของ property "eats" จะต้องเป็น instances ที่มาจากการ class "Plant"

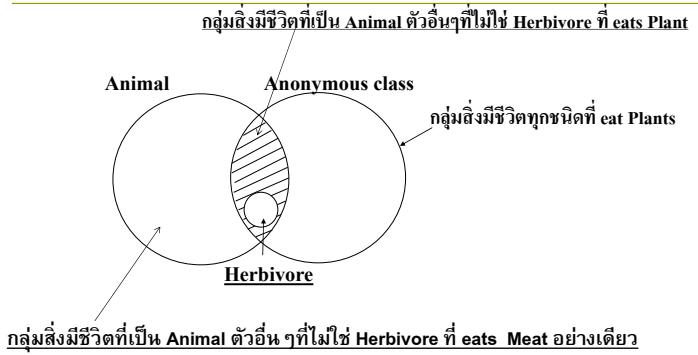
16



Herbivore is a subclass of an "Anonymous class"



Herbivore is a subclass of an "Anonymous class"



18



```
@base <http://www.animal.com/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ani: <http://www.animal.com/> .
```

```
# Define the classes
<Animal> a owl:Class .
<Herbivore> a owl:Class ;
rdfs:subClassOf <Animal> ;
rdfs:subClassOf _:x1.

_:x1 a owl:Restriction ;
owl:onProperty <eats> ;
owl:allValuesFrom <Plant>.

<Carnivore> a owl:Class ;
rdfs:subClassOf <Animal> ;
rdfs:subClassOf _:x2.

_:x2 a owl:Restriction ;
owl:onProperty <eats> ;
owl:allValuesFrom <Meat>.
```

animal.ttl

```

<Omnivore> a owl:Class;
rdfs:subClassOf <Animal>.

<Food> a owl:Class.
<Plant> a owl:Class ;
rdfs:subClassOf <Food> .
<Meat> a owl:Class ;
rdfs:subClassOf <Food> .

# Define the property eats
<eats> a owl:ObjectProperty ;
rdfs:domain <Animal> ;
rdfs:range <Food>.

ani:Cow
a <Herbivore>;
<eats> ani:Grass.

ani:Tiger
a <Carnivore>;
<eats> ani:Beef;
<eats> ani:Pork.

  
```

Instance level

19



Instance Level of Animal Ontology

ani:Cow
a <Herbivore>;
<eats> ani:Grass.

ani:Tiger
a <Carnivore>;
<eats> ani:Beef;
<eats> ani:Pork.

ถ้าเมื่อวานไม่ได้มีการบอกว่า "ani:Grass" เป็น instance ที่อยู่ใน class ไหน แต่ Inference engine ก็จะพิจารณาว่า ani:Cow เป็น instance ที่อยู่ใน class Herbivore และมีการใช้ property "eats" ดังนั้น สิ่งที่อยู่หลัง eats จะต้องเป็น instances ที่อยู่ใน class Plant เท่านั้น

ถ้าเมื่อวานไม่ได้มีการบอกว่า "ani:Beef" และ "ani:Pork" เป็น instance ที่อยู่ใน class ไหน แต่ Inference engine ก็จะพิจารณาว่า ani:Tiger เป็น instance ที่อยู่ใน class Carnivore และมีการใช้ property "eats" ดังนั้น สิ่งที่อยู่หลัง eats ทุกตัวจะต้องเป็น instances ที่อยู่ใน class Meat เท่านั้น

จาก Instance level ข้างบนนี้ เมื่อมีการประมวลผลร่วมกับ Schema level เครื่องมืออัตโนมัติ (inference engine) จะรู้ได้วันที่เลียบว่า "ani:Grass" ก็คือ "Plant" (เป็น instance ที่อยู่ใน class Plant) ส่วน "ani:Beef" และ "ani:Pork" ก็คือ "Meat" (เป็น instances ที่อยู่ใน class Meat)

20

Three forms of rdfs:subClassOf

① rdfs:subClassOf <Animal>

จะมี class หรือ rdfs:subClassOf

② rdfs:subClassOf
[a owl:Restriction ;
owl:onProperty <eats> ;
owl:allValuesFrom <Plant>
].

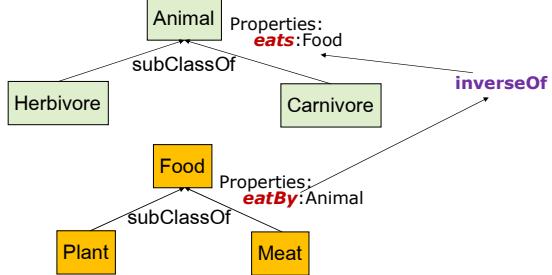
จะมี anonymous class หรือ rdfs:subClassOf

③ rdfs:subClassOf _:x1.
_:x1 a owl:Restriction ;
owl:onProperty <eats> ;
owl:allValuesFrom <Plant> .

21

Challenge!!

- ถ้าต้องการสร้าง property "eatBy" ให้เป็น inverse of "eats" property และกำหนดให้ "eatBy" มี domain คือ class "Food" และมี range คือ class "Animal"



22

Using owl:inverseOf Property

```

@base <http://www.animal.com/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ani: <http://www.animal.com/> .

.....
a owl:ObjectProperty;
rdfs:domain <Animal>;
rdfs:range .....

.....
a owl:ObjectProperty;
owl:inverseOf .....
  
```

23

Using owl:inverseOf Property

```

@base <http://www.animal.com/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ani: <http://www.animal.com/> .

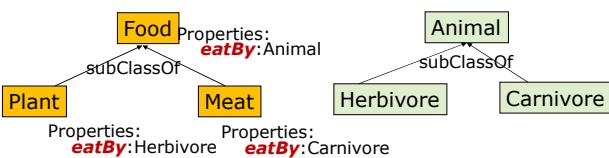
<eats>.....
a owl:ObjectProperty;
rdfs:domain <Animal>;
rdfs:range ...<Food>.....

<eatBy>.....
a owl:ObjectProperty;
owl:inverseOf .....<eats>.....
  
```

24

Challenge!!

- ถ้าต้องการกำหนดเพิ่มเติมว่า property "eatBy" ถ้าใช้กับ class "Plant" จะต้องมี range คือ class "Herbivore" แต่ถ้าหากใช้กับ class "Meat" จะต้องมี range เป็น class "Carnivore" จะสามารถออกแบบ OWL ได้อย่างไร



25

Using owl:allValuesFrom for eatBy

```

.....a owl:Class ;
rdfs:subClassOf <Food>.
rdfs:subClassOf
[ a owl:Restriction ;
owl:onProperty <eatBy> ;
.....
].
  
```

26

Using owl:allValuesFrom for eatBy

```
.....<Plant>.....a owl:Class ;
rdfs:subClassOf <Food>.
rdfs:subClassOf
[ a owl:Restriction ;
owl:onProperty <eatBy> ;
owl:allValuesFrom <Herbivore>.....].
]
```

27

Using owl:allValuesFrom for eatBy

```
.....a owl:Class ;
rdfs:subClassOf <Food>.
rdfs:subClassOf
[ a owl:Restriction ;
owl:onProperty <eatBy> ;
.....].
]
```

28

Using owl:allValuesFrom for eatBy

```
.....<Meat>.....a owl:Class ;
rdfs:subClassOf <Food>.
rdfs:subClassOf
[ a owl:Restriction ;
owl:onProperty <eatBy> ;
owl:allValuesFrom <Carnivore>.....].
]
```

29

Using OWL Class to change the local scope of property

owl:someValuesFrom

Understanding owl:someValuesFrom

```
<Professor>
rdfs:subClassOf <AcademicStaff>.

<Professor> owl:equivalentClass
[ a owl:Restriction ;
owl:onProperty <supervises> ;
owl:someValuesFrom <GraduateStudent> ] .
```

โดยนี้จะอ่านว่า Professor class จะเป็น subClassOf AcademicStaff class และ Professor class ก็ยัง equivalent กับ class ที่เป็น Anonymous class ซึ่งเป็น class ที่มี property supervises ซึ่ง value ของ property supervises จะเป็นชื่อมูลบางตัว (some values) หรืออย่างน้อย 1 ตัว (at least one) ต้องเป็น instance ที่อยู่ใน class GraduateStudent
▶ หรืออ่านให้เข้าใจคือ Professor class เป็น subClassOf AcademicStaff ซึ่ง Professor class จะต้องมีการใช้ property supervises อย่างน้อย 1 property ที่มีค่าชื่อมูลของ property นั้น เป็น instance ของ GraduateStudent."

37

An instance Level

```
per:49877888 a <UnderGraduateStudent>.
per:48977887 a <GraduateStudent>.
per:47688996 a <UnderGraduateStudent>.
per:ST949318 <supervises> per:49877888,
per:48977887,
per:47688996.
```

Per:ST949318 ถ้ามีการใช้ property supervises ค่าชื่อมูลของ supervises จะต้องมีชื่อมูลอย่างน้อย 1 ตัวที่เป็น instance ที่อยู่ใน class GraduateStudent!
(ชื่อมูลตัวอื่นของ supervises อาจไม่ได้เป็น GraduateStudent ก็ได้)

ดังนั้น ถ้าเราถาม inference engine ว่า per:ST949318 จะมี rdf:type เป็นอะไร inference engine ก็จะตอบว่า "Professor" และ "AcademicStaff"

38

allValuesFrom vs. someValuesFrom

```
[ a owl:Restriction ;
  owl:onProperty <supervises> ;
  owl:allValuesFrom <GraduateStudent>
] .
```

เมื่อไรก็ตามที่ class มีการใช้ property `supervises`, ค่าข้อมูลของ `supervises` จะต้องเป็น instances ของ `GraduateStudent`. แต่ก็ไม่บังคับว่า class นั้น จะต้องใช้ `supervises` ทุกครั้ง (ไม่ใช่ได้)

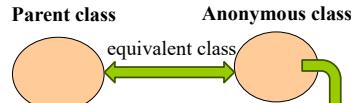
```
[ a owl:Restriction ;
  owl:onProperty <supervises> ;
  owl:someValuesFrom <GraduateStudent>
] .
```

บังคับว่า class จะต้องมีการใช้ property `supervises` อย่างน้อย 1 property ที่มีค่าข้อมูล เป็น instance ที่อยู่ใน `GraduateStudent` ด้วย ซึ่ง class จะไม่มีการใช้ `supervises` ไม่ได้

39

Another Example of using owl:someValuesFrom

- ถ้าเราต้องการอธิบายว่า 'Parent' class จะเป็น class ของคนที่มีลูก (child) อย่างน้อย 1 คนขึ้นไป



```
:Parent owl:equivalentClass
[ rdf:type owl:Restriction ;
  owl:onProperty :hasChild ;
  owl:someValuesFrom :Person
] .
```

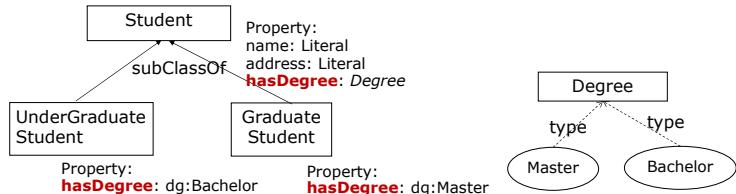
สามารถอธิบาย `Anonymous` class นี้เป็น class ของคนที่มี "hasChild" property อย่างน้อย 1 คนที่เป็น instance อยู่ใน class `Person`

40

Using OWL Class to change the local scope of property

`owl:hasValue`

Example: All UnderGraduate students have degree "Bachelor" value



UnderGraduateStudent class จะสืบทอด property "hasDegree" จาก class Student และถ้าเราต้องการกำหนดเพิ่มเติมว่าถ้าใครก็ตามที่เป็น UnderGraduateStudent จะมี property `hasDegree` ที่มีค่าข้อมูลเป็น instance "Bachelor" ที่อยู่ใน class Degree ส่วนใครก็ตามที่เป็น GraduateStudent ก็จะกำหนดให้มี property `hasDegree` ที่มีค่าข้อมูล เป็น instance "Master" ที่อยู่ใน class Degree แทน เราจะทำได้อย่างไร?

Defining the "hasDegree" property to have the value Bachelor (when used in UnderGraduateStudent)

```
@base <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dg: <http://www.university.edu/degree/> .
@prefix per: <http://www.university.edu/person/> .

# Define Class
<UnderGraduateStudent> a owl:Class ;
  rdfs:subClassOf <Student>.

<UnderGraduateStudent> owl:equivalentClass
  [ a owl:Restriction ;
    owl:onProperty <hasDegree> ;
    owl:hasValue dg:Bachelor
  ] .
```

Use Protégé to create this ontology too.

43

Defining the "hasDegree" property to have the value Bachelor (when used in UnderGraduateStudent)

```
<GraduateStudent> a owl:Class ;
  rdfs:subClassOf <Student>.

<GraduateStudent> owl:equivalentClass
  [ a owl:Restriction ;
    owl:onProperty <hasDegree> ;
    owl:hasValue dg:Master
  ] .
per:name a owl:DatatypeProperty;
  rdfs:domain <Student> ;
  rdfs:range rdfs:Literal.

per:address a owl:DatatypeProperty;
  rdfs:domain <Student> ;
  rdfs:range rdfs:Literal.
```

```
dg:Bachelor
  a <Degree>.
dg:Master
  a <Degree>.

per:49778888
  a <UnderGraduateStudent>;
  per:name "Willy John";
  per:address "Chicago, USA".

per:54822442
  a <GraduateStudent>;
  per:name "Robert Lee";
  per:address "New York, USA".
```

44



Understanding owl:hasValue

```
<UnderGraduateStudent> a owl:Class ;
  rdfs:subClassOf <Student>.
<UnderGraduateStudent> owl:equivalentClass
  [ a owl:Restriction ;
    owl:onProperty <hasDegree> ;
    owl:hasValue dg:Bachelor ] .
```

Note that this is an instance of the class **Degree**.

ด้วยนี่จะอ่านว่า UnderGraduateStudent class จะเป็น subClassOf Student และ UnderGraduateStudent ที่ยังเป็น equivalentClass กับ Anonymous class ซึ่งเป็น class ที่มี property - **hasDegree** ที่มีค่าข้อมูลคือ instance Bachelor ที่อยู่ใน class Degree หรือถ้าให้ภาษาขึ้นคือ คณทุกคนที่เป็น UnderGraduateStudent จะต้องมี **hasDegree** property ซึ่งมีค่าข้อมูลเป็น Bachelor

45



An instance of UnderGraduateStudent

```
@base <http://www.university.edu/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dg: <http://www.university.edu/degree/> .
@prefix per: <http://www.university.edu/person/> .
```

```
.....  
per:49778888  
a <UnderGraduateStudent>;  
per:name "Willy John";  
per:address "Chicago, USA";  
<hasDegree> dg:Bachelor.
```

Instances ทุกด้านของ class UnderGraduateStudent จะต้องมี Property "hasDegree" ซึ่งมีค่าเป็น Bachelor หมายเหตุ: เราไม่จำเป็นต้องสร้าง property hasDegree ให้กับ instances ของ UnderGraduateStudent ในเอกสารนี้ เพราะ inference engine จะทำการสร้าง property "hasDegree" ที่มีค่าข้อมูลเป็น instance dg:Bachelor ให้โดยอัตโนมัติ จากคุณสมบัติของ hasValue นั้นเอง ซึ่งสรุปง่ายๆ ก็คือ inference engine จะรู้ด้วยตนเองว่า property ที่เป็น UnderGraduateStudent ก็จะมี degree เป็น Bachelor นั้นเอง



Triples to be generated by Inference Engine

```
(per:49778888 , rdf:type, UnderGraduateStudent)
(per:49778888 , per:name, "Willy John")
(per:49778888 , per:address, "Chicago, USA")
(per:49778888 , <hasDegree>, dg:Bachelor)
```

This statement has been inserted automatically.

47



Summary of the different ways a class can constrain a property

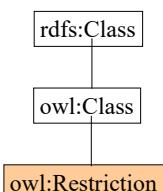
□ ในสไลด์ก่อนหน้านี้ เราได้เห็นวิธีการต่างๆ ที่จะกำหนดเงื่อนไขให้กับ class ที่มี การใช้ property ที่เรากำหนดขึ้น ว่าสำหรับ class นั้นการใช้ property ด้านนี้ เราจะ ควบคุมให้มีการเปลี่ยนค่าข้อมูลของ property นั้นเป็นแบบใดได้บ้าง โดยอาศัย คุณสมบัติดังต่อไปนี้:

- ค่าข้อมูลทุกด้านของ property นั้นจะเป็น instances ที่อยู่ใน class ที่เรากำหนด ขึ้นโดยเฉพาะ โดยการใช้คุณสมบัติ **owl:allValuesFrom**
- ค่าข้อมูลของ property นั้นอย่างน้อยหนึ่งด้านจะต้องเป็น instance ที่อยู่ใน class ที่เรากำหนดขึ้นโดยเฉพาะ โดยการใช้คุณสมบัติ **owl:someValuesFrom**
- ค่าข้อมูลของ property นั้นจะต้องมีค่าเป็น instance ที่สามารถกำหนดขึ้น ได้โดยเดียว ที่อยู่ใน class ใด class หนึ่ง โดยอาศัยคุณสมบัติ **owl:hasValue**

48



Properties of the Restriction Class



Properties:

onProperty: *rdf:Property*
allValuesFrom: *rdfs:Class*
someValuesFrom: *rdfs:Class*
hasValue: <instance value>

49

Using OWL Class to define cardinality constraints

owl:cardinality

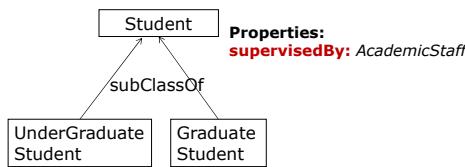
Context-specific cardinality constraints

□ **Definition of cardinality:** หมายถึงจำนวนการเกิดขึ้น

□ ต่อไปนี้เรามาดูวิธีการจำกัดจำนวน property เมื่อมีการใช้ property นั้นกับ class ใด class หนึ่งดังต่อไปนี้

52

An undergraduate student can have only one supervisedBy property (cardinality = 1)



ถ้าต้องการกำหนด class **UnderGraduateStudent** สามารถใช้ property **supervisedBy** (ที่สืบทอดจาก class **Student**) ได้เพียง property เดียวเท่านั้น หมายความว่า นักศึกษาระดับปริญญาตรี (**UnderGraduateStudent**) แต่ละคน จะถูก **supervisedBy** อาจารย์ที่ปรึกษา (**AcademicStaff**) เพียงแค่ 1 คนเท่านั้น (ใช้ property **supervisedBy** ได้เพียง 1 ครั้ง) เราจะมีวิธีการกำหนดได้อย่างไร?

Defining the cardinality of the supervisedBy property to be 1

```

@base <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix per: <http://www.university.edu/person/> .

# Define Class
<Student> a owl:Class.

<UnderGraduateStudent> a owl:Class ;
  rdfs:subClassOf <Student>.

<UnderGraduateStudent> owl:equivalentClass
[ a owl:Restriction ;
  owl:onProperty <supervisedBy> ;
  owl:cardinality "1"^^xsd:int
] .
  
```

Use Protégé to create this ontology too.

```

# Define Property
<supervisedBy>
  a owl:ObjectProperty;
  rdfs:domain <Student>;
  rdfs:range <AcademicStaff>.

# Define Instance
per:49778888
  a <UnderGraduateStudent>;
  <supervisedBy> per:ST949318.
  
```

54

Defining the cardinality of the supervisedBy property to be 1

```

<UnderGraduateStudent> owl:equivalentClass
[ a owl:Restriction ;
  owl:onProperty <supervisedBy> ;
  owl:cardinality "1"^^xsd:int
] .
  
```

Each student must have only one supervisor

- ข้อความนี้อ่านว่า: class **UnderGraduateStudent** มีความเทียบเท่ากับ **Anonymous class** ที่มีการใช้ property **supervisedBy** เพียง property เดียว นั่นหมายความว่า นักศึกษาระดับปริญญาตรี (**UnderGraduateStudent**) จะสามารถถูก **supervisedBy** โดยอาจารย์ เพียงคนเดียว
- วิธีอ่านข้อความนี้ที่ง่ายกว่าคือ: class **UnderGraduateStudent** จะใช้ property **supervisedBy** ได้เพียงหนึ่งเดียวเท่านั้น

Defining the cardinality of the supervisedBy property to be 1

```

# Define Instance
per:49778888
  a <UnderGraduateStudent>;
  <supervisedBy> per:ST949318.
  
```

The 49778888 has only one supervisedBy.

นักศึกษารหัส **per:49778888** ถ้ามี **rdf:type** เป็น **UnderGraduateStudent** จะสามารถใช้ property **supervisedBy** ได้เพียงตัวเดียวเท่านั้น

56

Defining the cardinality of the supervisedBy property to be 1

```

# Define Instance
per:49778888
  a <UnderGraduateStudent>;
  <supervisedBy> per:ST949318;
  <supervisedBy> per:DavidLee;
} sameAs
  
```

แต่ถ้ามีการใช้ property **supervisedBy** มากกว่าหนึ่งตัว เช่นจากตัวอย่างข้างบน นักศึกษารหัส **per:49778888** จะถูก **supervisedBy** อาจารย์ 2 คน คือ **per:ST949318** และ **per:DavidLee**

➔ กรณีนี้ inference engine จะทำการอนุญาตให้ว่า **ST949318** จะต้องเป็นคนๆเดียวกับ **DavidLee** (**ST949318 sameAs ST949318**)



owl:minCardinality and owl:maxCardinality

- ถ้าต้องการอธิบายว่า Pizza แต่ละชิ้นจะมี toppings ได้อย่างน้อย 1 topping และมี toppings ได้มากที่สุด 3 toppings เราจะอธิบายได้อย่างไร?

```
<Pizza> owl:equivalentClass
[ a owl:Restriction ;
  owl:onProperty <hasToppings> ;
  owl:minCardinality "1"^^xsd:int;
  owl:maxCardinality "3"^^xsd:int
] .
```

58



owl:minCardinality and owl:maxCardinality

Use Protégé to create this ontology too.

There are 3 kinds of topping (greater than 2 toppings).

```
@base <http://www.pizza.com/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
<Pizza> owl:equivalentClass
[ a owl:Restriction ;
  owl:onProperty <hasToppings> ;
  owl:maxCardinality "2"^^xsd:int
].
<hasToppings> a owl:DatatypeProperty;
rdfs:domain <Pizza>;
rdfs:range xsd:string.
```

```
<Seafood>
a <Pizza>;
<hasToppings> "sausage", "onion", "black oliver".
```

The inference engine does not allow <Seafood> pizza to have topping more than 2 toppings.

59



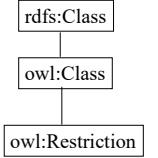
Summary of the different ways to express the cardinality of a property

- ในสไลด์ก่อนหน้านี้ เราได้เห็นวิธีการระบุจำนวน property ให้กับ class โดยการใช้คุณสมบัติดังต่อไปนี้
 - owl:cardinality
 - owl:minCardinality
 - owl:maxCardinality

60



Complete List of Properties of the Restriction Class



Properties:

- onProperty: rdf:Property
- allValuesFrom: rdfs:Class
- someValuesFrom: rdfs:Class
- hasValue: <Instance>
- cardinality: xsd:nonNegativeInteger
- minCardinality: xsd:nonNegativeInteger
- maxCardinality: xsd:nonNegativeInteger

61

Using OWL to set equivalent Property

owl:equivalentProperty

Subject name is equivalent to the Title property in Dublin Core



Subject

Properties:
name: Literal

ถ้าต้องการกำหนดให้ **property "name"** มีความหมายเหมือนกับ **property "dcterms:title"** เราจะกำหนดได้อย่างไร?



Defining name to be equivalent to dcterms:title

@base <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .

<name> a owl:DatatypeProperty;
owl:equivalentProperty dcterms:title;
rdfs:domain <Subject>;
rdfs:range rdfs:Literal.

dcterms:title a owl:DatatypeProperty.

Use Protégé to
create this
ontology too.

ในดู. นั่งกำหนดให้ instance <CS10001> ที่เป็น Subject มีการใช้ property <name> อย่างไรก็ตาม เครื่องเรียนรู้สามารถ inference engine (โดยใช้ SPARQL) ว่า <CS10001> มี dcterms:title คืออะไรได้ ยัง เช่น ผลลัพธ์การ inference ของ <name> เป็น <value> นั่นเอง

Use Protégé to
create this
ontology too.

```
<CS10001>
  a <Subject>;
  <name> "Semantic Web".
```

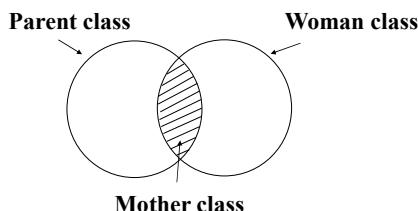
Using OWL to manage Classes

owl:intersectionOf
owl:disjointWith
owl:unionOf
owl:oneOf



Understanding owl:intersectionOf

- ▢ ถ้าเราต้องการกำหนดให้ class Mother เกิดจากการ intersection กันระหว่าง class Parent กับ class Woman เราจะสร้างด้วย OWL ได้อย่างไร



1

 Semantic Web

Understanding owl:intersectionOf

```
@prefix : <http://example.com/owl/person/> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
@prefix owl: <http://www.w3.org/2002/07/owl#> .
```

:Woman a owl:Class.
:Parent a owl:Class.

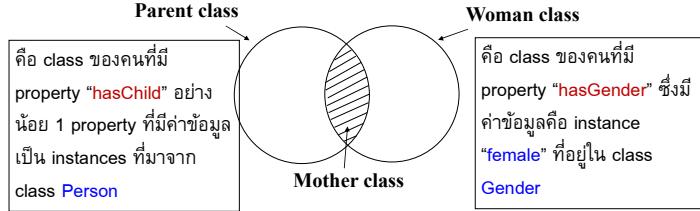
:Mother a owl:Class;
owl:intersectionOf (:Woman :Parent) .

```
:3168812422
  a :Woman.
:3132234433
  a :Woman.
:3168812422
  a :Parent.
```

ສູ່ທ່ຽນດ້ວຍຢ່າງນີ້ເມື່ອເຮົາສຸກຄຸນກົມ inference engine (ໂດຍໃຊ້ SPARQL) ເກີຍກັບ `rdf:type` ຂອງຄົນທີ່ມ່ຽຮສົກ :3168812422 ກັຈະໄດ້ຄຳຕອບຄືອື່ນ class "Woman", "Parent" ແລະ "Mother"

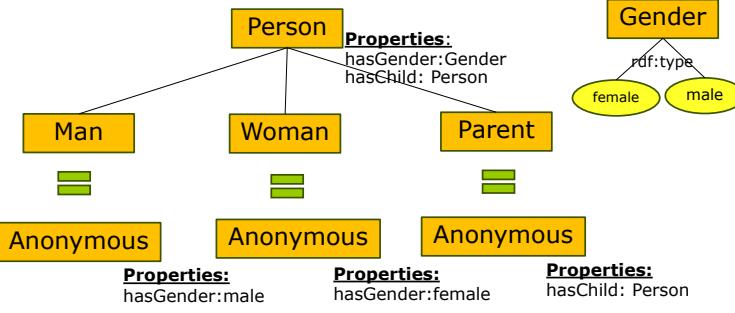
Another example of owl:intersectionOf

ถ้าต้องการอธิบายว่า class Mother เกิดจากการ intersection ของ class Parent และ class "Woman" โดย class Parent ก็คือคนทุกคนที่มี property "hasChild" อย่างน้อย 1 property ซึ่งค่าข้อมูลของ property นี้จะเป็น instance ที่เป็นคนที่อยู่ใน class Person นั้นเอง และ class Woman ก็จะได้แก่คนทุกคนที่มี property "gender" ซึ่งมีค่าข้อมูลคือ "female"



11

Another example of owl:intersectionOf



12

Understanding owl:intersectionOf

```

@prefix : <http://example.com/owl/person/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
  
```

```

:Person a owl:Class.
:Gender a owl:Class.
:Woman
  rdfs:subClassOf :Person;
  owl:equivalentClass _:x1.
  _:x1
    a owl:Restriction;
    owl:onProperty :hasGender;
    owl:hasValue :female.
:Man
  rdfs:subClassOf :Person;
  owl:equivalentClass _:x2.
  _:x2
    a owl:Restriction;
    owl:onProperty :hasGender;
    owl:hasValue :male.
  
```

Use Protégé to create this ontology too.

```

:Parent
  rdfs:subClassOf :Person;
  owl:equivalentClass _:x3.
  _:x3 a owl:Restriction;
  owl:onProperty :hasChild;
  owl:someValuesFrom :Person.
:Mother a owl:Class;
  owl:intersectionOf (:Parent :Woman) .

:hasGender a owl:ObjectProperty;
  rdfs:domain :Person;
  rdfs:range :Gender.
:hasChild a owl:ObjectProperty;
  rdfs:domain :Person;
  rdfs:range :Person.
  
```

Understanding owl:intersectionOf

Define Instances

```

:female a :Gender.
:male a :Gender.

:David a :Person.
:Willy a :Person.
:Mary a :Person.
:Jane a :Person.
:Katty a :Person.
:Rose a :Person.

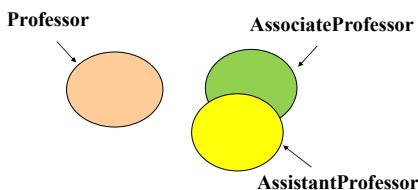
:Mary :hasChild :David.
:Jane :hasGender :female.
:Mary :hasGender :female.
:Katty :hasGender :female.
:Rose :hasChild :Katty.
:David :hasChild :Jane.
:Rose :hasGender :female.
  
```

Use Protégé to create this ontology too.

ให้ดูบอร์ดว่ามีโครงสร้างที่มี rdf:type เป็น Woman, Man, Parent, หรือ Mother

Understanding owl:disjointWith

ถ้าต้องการอธิบายว่า class Professor จะ disjoint กับ class AssociateProfessor และ AssistantProfessor เราจะเขียน owl ได้อย่างไร



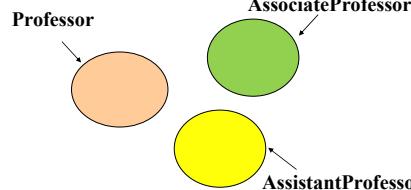
:Professor owl:disjointWith :AssociateProfessor,
:AssistantProfessor.

จากที่จะสามารถบอกได้ว่าคนหนึ่งที่คือ อะดัตจานไม่มี instances ใดๆ ของ class Professor ไปเป็นสมาชิกอยู่ใน class AssociateProfessor หรือ AssistantProfessor อย่างไรก็ตาม ที่ไม่ได้มีความคล้ายคลึงกันมากเท่าๆ กัน classes นี้จะ disjoint กันทั้งหมด เพราะ class AssociateProfessor ก็อาจมีสมาชิกที่เป็น instances ที่อยู่ใน AssociateProfessor และ AssistantProfessor ด้วยกันทั้งสี่ได้

15

Understanding owl:disjointWith

ดังนั้น ถ้าต้องการบังคับให้ทั้งสาม classes นี้มีการ disjoint กันทั้งหมด คือต้องไม่มีสมาชิกร่วมกันเลย เราจะต้องอธิบายว่าทั้งสามนี้ต่างกันเป็น disjoint กันทุกๆ classes



```

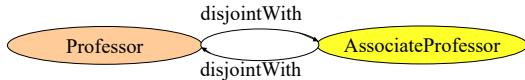
:Professor a owl:Class.
:AssociateProfessor a owl:Class;
  owl:disjointWith :Professor.
:AssistantProfessor a owl:Class;
  owl:disjointWith :Professor;
  owl:disjointWith :AssociateProfessor.
:David a :Professor.
:Willy a :Professor.
:Mary a :AssociateProfessor.
:Jane a :AssistantProfessor.
:Mary a :Professor. ✗
  
```

Inference engine จะไม่อนุญาตให้มีการกำหนดให้ instance หนึ่งๆ (:Mary) เป็นสมาชิกอยู่ใน class มากกว่า 1 class

16

Note: disjointWith is a SymmetricProperty!

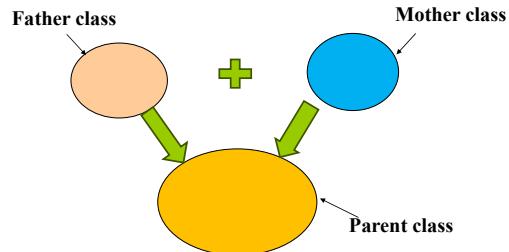
- Example: if Professor is disjointWith AssociateProfessor, then AssociateProfessor is disjointWith Professor.



17

Understanding owl:unionOf

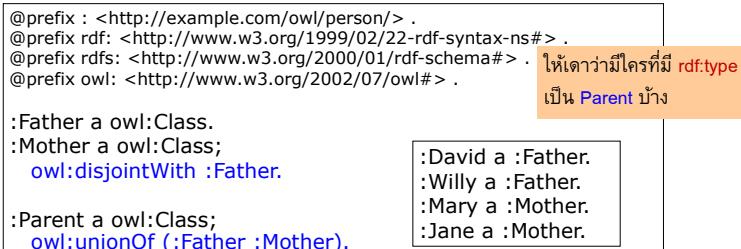
- ต้องการเขียนว่า class Parent เกิดจากการ union กันของ class Father และ class Mother โดย class Father นั้นจะต้อง disjoint กับ class Mother ด้วย



18

Understanding owl:unionOf

Use Protégé to create this ontology too.



19

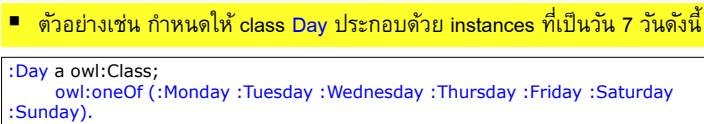
Understanding owl:oneOf

- จุดประสงค์ของการใช้ owl:oneOf คือการระบุว่า instances ตัวไหนบ้างที่เป็นสมาชิกของ class ที่เรากำหนด
- คำสั่งนี้จะมีประโยชน์เมื่อเราต้องการที่จะกำหนดให้ class ได้ class หนึ่งมีจำนวนสมาชิกตามที่เรากำหนดไว้เท่านั้น โดยมีการกำหนดจำนวน instances ที่ชัดเจนไปเลย (fixed set of members) และจะต้องไม่มี instances อื่นที่นอกเหนือจากน้อยอยู่ใน class นั้น

20

Understanding owl:oneOf

Use Protégé to create this ontology too.



- ตัวอย่างเช่น กำหนดให้ class Day ประกอบด้วย instances ที่เป็นวัน 7 วันดังนี้

```
:Day a owl:Class;
owl:oneOf (:Monday :Tuesday :Wednesday :Thursday :Friday :Saturday :Sunday).
```

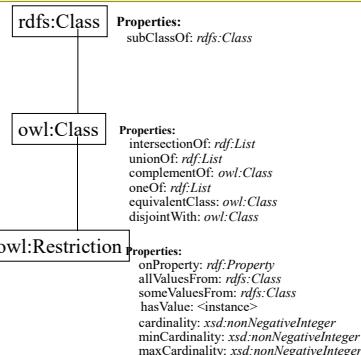
- หรือ class BobsChildren จะเป็น subclass of class Person และจะมีสมาชิกคือคน 3 คนต่อไปนี้เท่านั้น (ซึ่งเป็นลูกของ Bobs)

```
:Person a owl:Class.
:BobsChildren a owl:Class;
rdfs:subClassOf :Person;
owl:oneOf (:Bill :John :Mary).
```

สำหรับ inference engine ว่า :Bill (หรือ :John หรือ :Mary) มี rdf:type (เป็นอะไร) ที่จะตอบว่า :BobsChildren นั้นเอง

21

Summary of Class Properties



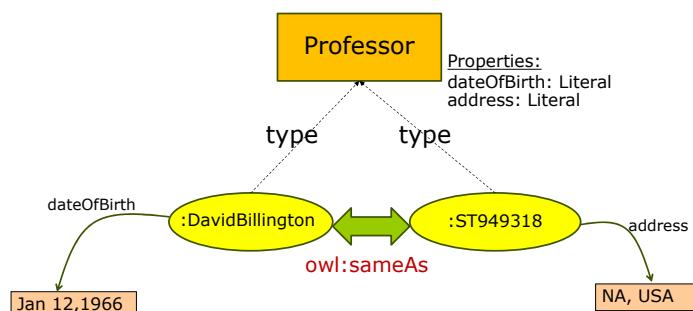
22

OWL statements that you can incorporate into instances

owl:sameAs
owl:differentFrom
owl:allDifferent



Indicating that two instances are the same



27

Indicating that two instances are the same

Use Protégé to create this ontology top.

```

@prefix : <http://www.university.edu/staff/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:Professor a owl:Class.
:dateOfBirth a owl:DatatypeProperty;
  rdfs:domain :Professor;
  rdfs:range rdfs:Literal.
:address a owl:DatatypeProperty;
  rdfs:domain :Professor;
  rdfs:range rdfs:Literal.
  
```

```

:DavidBillington
a :Professor;
:dateOfBirth "Jan 12,1966".

:ST949318
a :Professor;
:address "NA, USA";
owl:sameAs :DavidBillington.
  
```

สามารถใช้คำสั่ง owl:sameAs ในระดับของ instance level เพื่อระบุว่า instance :ST949318 จะเหมือนกับ instance :DavidBillington (คนสองคนนี้เป็นคนเดียวกัน)



owl:sameAs

```

:DavidBillington
a :Professor;
:dateOfBirth "Jan 12,1966".

:ST949318
a :Professor;
:address "NA, USA";
owl:sameAs :DavidBillington.
  
```

```

:DavidBillington
a :Professor;
:dateOfBirth "Jan 12,1966";
:address "NA, USA".

:ST949318
a :Professor;
:address "NA, USA";
:dateOfBirth "Jan 12,1966";
  
```

การใช้คุณสมบัติ owl:sameAs จะทำให้ inference engine สามารถ推断ได้ว่าคนสองคนนี้เป็นคนๆเดียวกัน ดังนั้น ถ้าแต่ละคนมีกำหนด properties ต่างๆที่เป็นของตนเอง Properties เหล่านี้จะถูกบูรณาเว้นเข้าด้วยกัน เพราะถือว่าเป็นคนๆเดียวกัน ดังนั้น เราจึงสามารถ query หา date of birth ของคนที่ตัว :ST949318 ได้หรือ query หา address ของคนชื่อ :DavidBillington ได้ด้วยเช่นกัน

Indicating that two instances are different

```

:DavidBillington
a :Professor;
:dateOfBirth "Jan 12,1966".

:ST949318
a :Professor;
:address "NA, USA";
owl:differentFrom :DavidBillington.
  
```

ในการนี้ที่เราต้องการกำหนดว่า instance :ST949318 มีความแตกต่างกับ instance :DavidBillington จะสามารถทำได้โดยใช้คำสั่ง owl:differentFrom



owl:AllDifferent

■ เราสามารถใช้คุณสมบัติ owl:AllDifferent class เพื่อกีบ instances ทุกดัวที่มีความแตกต่างกัน

```

:DavidBillington a :Professor.
:DavidBill a :Professor.
:BillDavid a :Professor.
  
```

ด้านนี้เป็นการกำหนด blank node ขึ้นมาใหม่ type เมื่อ owl:AllDifferent และใช้ owl:distinctMembers เพื่อระบุว่า Instances ต่อไปนี้เดียวกัน :DavidBillington, :DavidBill และ :BillDavid ต่างกันมีความแตกต่างกันทุกดัว

```

:_x1 a owl:AllDifferent;
owl:distinctMembers (:DavidBillington :DavidBill :BillDavid).
  
```



31

Summary of the different statements you can incorporate into instances

▫ สรุปว่าคำสั่งต่อไปนี้ จะเป็นคำสั่งที่สามารถถูกใช้อยู่ภายใต้ instance level ได้เลย

- owl:sameAs
- owl:differentFrom
- owl:AllDifferent

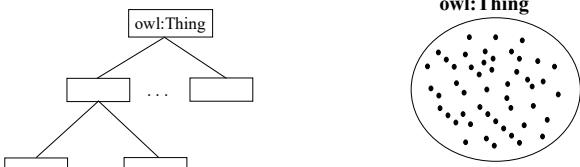
32

The owl:Thing class is the root of all classes

owl:Thing

The owl:Thing class is the root of all classes

▫ owl:Thing จะเป็นที่รวมของ classes ทุก classes ที่เราสร้างขึ้นมา ซึ่ง classes ทุก classes นี้จะต้องเป็น subclass ของ owl:Thing ทั้งหมด ดังนั้น instances ของทุก classes ก็จะเป็นสมาชิกของ owl:Thing ด้วย



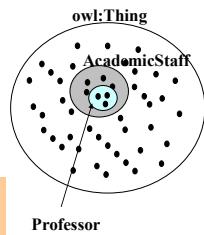
ทุกๆ instances ในทุก classes จะเป็น instances ของ owl:Thing!

34

owl:Thing

```
@base <http://www.university.edu/staff/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:Professor a owl:Class;
rdfs:subClassOf :AcademicStaff.
```



Professor

Class Professor และ class AcademicStaff ดังนี้ถูกอนุมานว่า เป็น subclasses ของ owl:Thing ด้วย

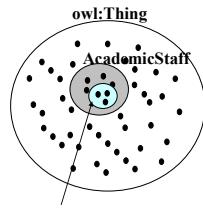
35

owl:Thing

```
@base <http://www.university.edu/staff/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:Professor a owl:Class;
rdfs:subClassOf :AcademicStaff.
```

:ST949318 a :Professor.



ดังนั้น ถ้า ST949318 มี type เป็น Professor, instance ST949318 ก็จะเป็น instance ของ AcademicStaff และเป็น instance ของ owl:Thing ด้วย

36

Importing other OWL documents

owl:Ontology
owl:imports



owl:Ontology Class

- owl:Ontology class จะเป็นอีก class ที่ถูกสร้างขึ้นมาเพื่อใช้อธิบาย ontology โดยผ่านทาง properties ต่างๆ รวมไปถึงการ import ontology จากที่อื่นๆด้วย

owl:Ontology

Properties:

imports:
versionInfo:
priorVersion: *Ontology*
incompatibleWith: *Ontology*
backwardCompatibleWith: *Ontology*

Note: *Ontology* นี้จะหมายถึง OWL document เช่น university.ttl.

38



The Ontology Header

```
@prefix : <http://www.mydomain.com/person/> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix owl: <http://www.w3.org/2002/07/owl#> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
  
:person a owl:Ontology;  
    owl:imports <file:D:/SemanticWeb/Content/test/equivalentClass.ttl>.
```

```
:388888888  
a :Woman.  
:488888888  
a :Woman.
```

You can also specify physical URL such as:
<https://computing.kku.ac.th/staff/myfile.ttl>

ตัวอย่างนี้จะเป็นการ import "ไฟล์ชื่อ 'equivalentClass.ttl'" เข้ามาบังไฟล์ ontology ปัจจุบัน (person.ttl) ซึ่งจะเป็นการรวมไฟล์ ontologies ทั้งสองไฟล์นี้เข้าเป็นไฟล์เดียวกัน

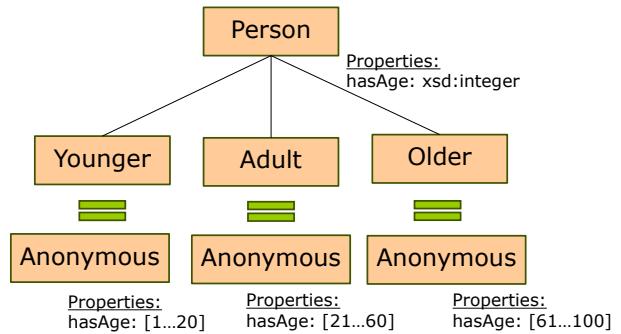
39

Additional Example

owl:DataRange
owl:onDatatype
owl:withRestrictions



Classifying Groups of Data



41



Classifying Groups of Data

```
@prefix : <http://www.example.com/person/> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
@prefix owl: <http://www.w3.org/2002/07/owl#> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
  
:Person a owl:Class.  
.hasAge a owl:DatatypeProperty;  
    rdfs:domain :Person;  
    rdfs:range xsd:integer.  
  
:Younger a owl:Class;  
    rdfs:subClassOf :Person;  
    owl:equivalentClass  
    [ a owl:Restriction;  
        owl:onProperty :hasAge;  
        owl:someValuesFrom  
        [ a owl:DataRange;  
            owl:onDatatype xsd:integer;  
            owl:withRestrictions ( [xsd:minInclusive 1] [xsd:maxInclusive 20] )  
        ];  
    ].
```

Use Protégé to
create this
ontology top.

42



Classifying Groups of Data

```
:Adult a owl:Class;  
    rdfs:subClassOf :Person;  
    owl:equivalentClass  
    [ a owl:Restriction;  
        owl:onProperty :hasAge;  
        owl:someValuesFrom  
        [ a owl:DataRange;  
            owl:onDatatype xsd:integer;  
            owl:withRestrictions ( [xsd:minInclusive 21] [xsd:maxInclusive 60] )  
        ];  
    ].  
  
:Older a owl:Class;  
    rdfs:subClassOf :Person;  
    owl:equivalentClass  
    [ a owl:Restriction;  
        owl:onProperty :hasAge;  
        owl:someValuesFrom  
        [ a owl:DataRange;  
            owl:onDatatype xsd:integer;  
            owl:withRestrictions ( [xsd:minInclusive 61] [xsd:maxInclusive 100] )  
        ];  
    ].
```

43

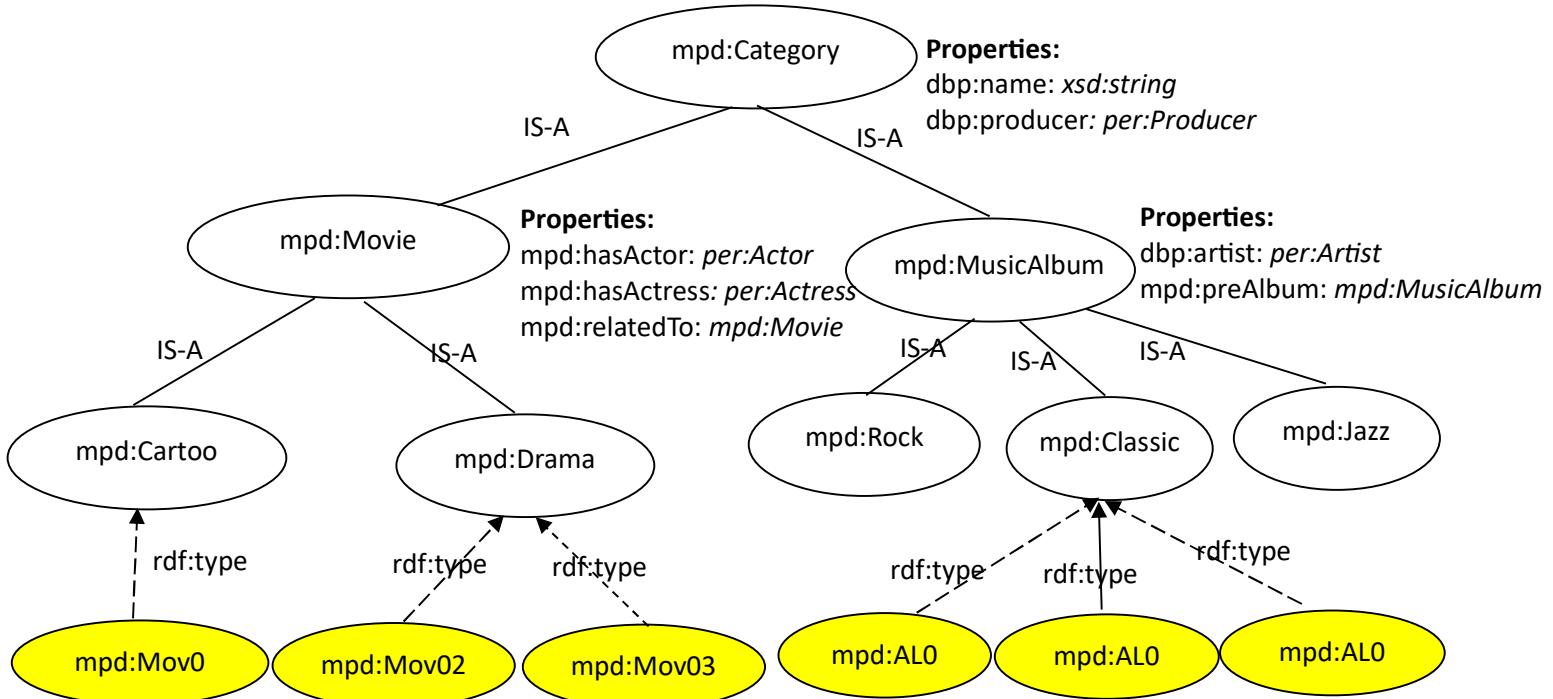
Classifying Groups of Data

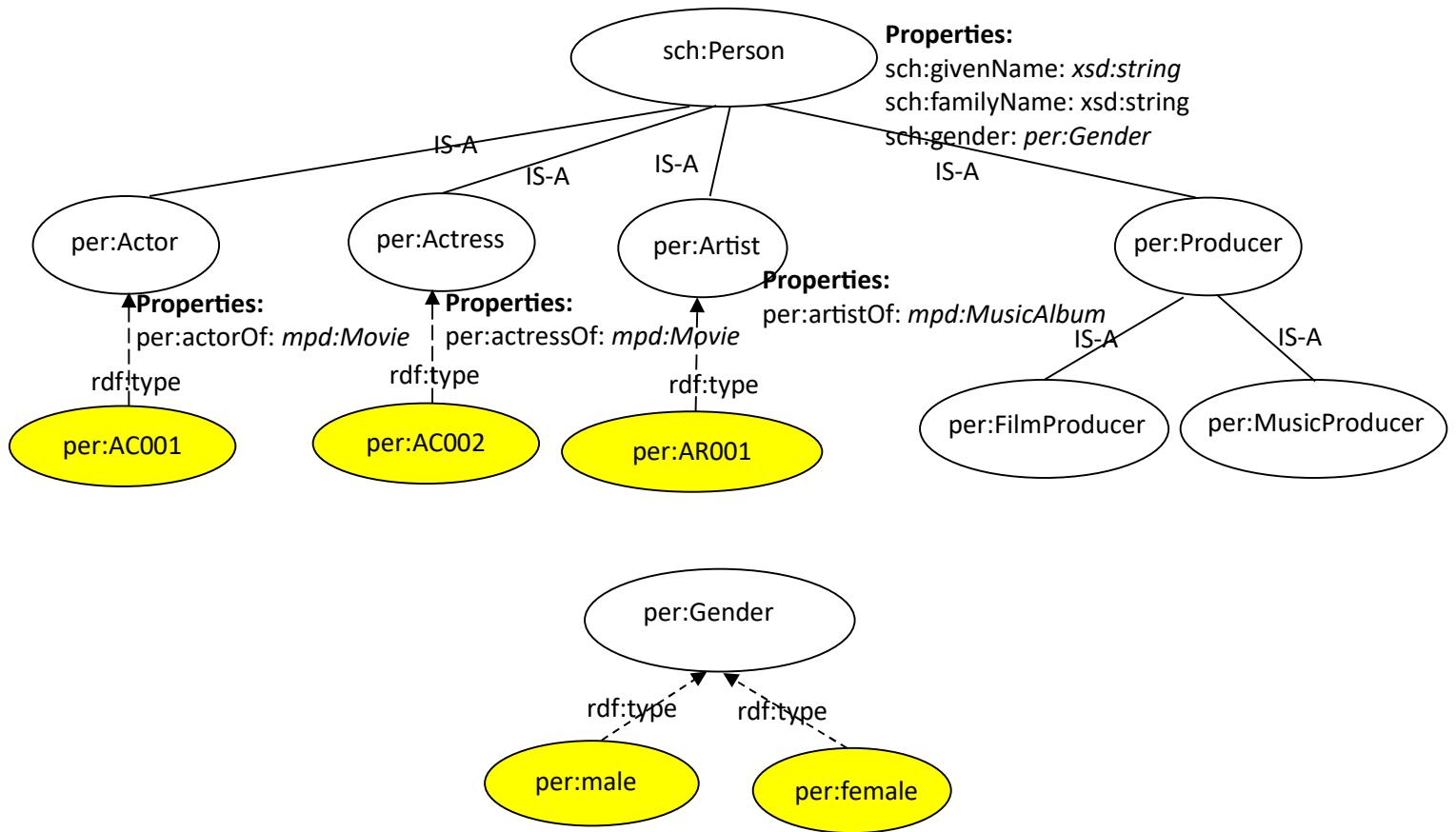
```
:David :hasAge "25"^^xsd:int.  
:Marry :hasAge "30"^^xsd:int.  
:John :hasAge "10"^^xsd:int.  
:Willy :hasAge "8"^^xsd:int.  
:Max :hasAge "61"^^xsd:int.  
:Bob :hasAge "90"^^xsd:int.  
:Smith :hasAge "40"^^xsd:int.
```

จากตัวอย่างนี้ จะทำให้เราสามารถ query ตาม inference engine ว่ามีใคร (instances ไหนบ้าง) ที่อยู่ class **Younger** (Willy, and John), มีใครบ้าง (instances ไหนบ้าง) ที่อยู่ใน class **Adult** (David, Marry, and Smith) และมีใครบ้าง (instances ไหนบ้าง) ที่อยู่ใน class **Older** (Max, and Bob) ซึ่งการ query จะใช้ **SPARQL** ที่เราจะได้เรียนในบทถัดไป

ตัวอย่างข้อสอบปลายภาค

1. จงทำเครื่องหมาย ✓ หรือ X หน้าข้อต่อไปนี้
 - 1.1 ถ้า Class B เป็น subclass ของ Class A แทนด้วย (B, subClassOf, A) ดังนั้น ถ้า (a, rdf:type, A) ก็จะได้ว่า (a, rdf:type, B) ด้วย
 - 1.2 ถ้า p เป็น symmetric property และถ้ากำหนดให้ (a, p, b) และ (b, p, c) และจะได้ว่า (a, p, c) ด้วย
 - 1.3 ถ้า (a, sameAs, b) และจะได้ว่า (b, sameAs, a) ดังนั้น sameAs เป็นคุณสมบัติ Transitive
 - 1.4 ถ้า (C1, rdf:type, owl:Class) และ (C2, rdf:type, owl:Class) และ ถ้า (p1, rdfs:domain, C1) และ (p1, rdfs:range, C2) จะได้ว่า (p1, rdf:type, owl:DatatypeProperty)
 - 1.5 ถ้า (p1, equivalentProperty, p2) และถ้า (a, p1, b) จะได้ว่า (a, p2, b) ด้วย
2. ให้ไปทบทวนเนื้อหาบทที่ 2.1, 2.2 และ 2.3 เกี่ยวกับ concept ของ LOD จะมีการเติมคำลงในช่องว่าง (โดยเลือกจากกลุ่มคำที่กำหนดไว้ให้) เช่น
 - 2.1 เป็นโปรโตคอลที่ใช้กับ Linked Open Data
 - 2.2 DBpedia คือตัวอย่างของ Linked Open Data ที่มีการเก็บข้อมูลในรูปแบบของ RDF ซึ่งอยู่ในฐานข้อมูลขนาดใหญ่ให้ผู้คนจากทั่วโลกเข้ามาใช้งานข้อมูลได้
 - 2.3 SPARQL เป็นภาษาสืบค้นข้อมูลที่เก็บในรูปแบบของ RDF
 - 2.4 Microdata RDF ถูกใช้ใน HTML5 เพื่อใช้ในการ annotate เว็บเพจ
 - 2.5 Schema.org เป็น vocabulary หนึ่งที่เกิดจากความร่วมมือกันของ Bing, Google, และ Yahoo
3. จากตัวอย่างดอนโน่โดยต่อไปนี้





3.2 ถ้ามีการกำหนดดังต่อไปนี้

(mpd:relatedTo, rdf:type, owl:SymmetricProperty)

และถ้ามีการสร้าง instance level ดังต่อไปนี้ และนำไปประมวลผลกับคอมพิวเตอร์ พร้อมกับอนโนทेशันโดยข้างต้น

(mpp:Mov02, mpd:relatedTo, mpd:Mov03)

3.2.1 จงแสดงประโยชน์ในระดับ instance level ที่อยู่ในรูปแบบของ triple ที่คอมพิวเตอร์จะสร้างหรืออนุมานขึ้นมาให้

(3 คะแนน) **(mpd:03 mpd:relatedTo mpd:02)**
คำตอบ.....

3.2.2 ให้เขียน SPARQL เพื่อสืบค้นหาว่ามีรัศสภาพยน์เรื่องไฟหนึ่งที่มีความสัมพันธ์(relatedTo) กันบ้าง (5 คะแนน)

คำตอบ

PREFIX mpd : <http://www.mydomain.com/>

SELECT *

WHERE {

mpd : Movie mpd:relatedTo ?SymmetricProperty

}

มอย mpd:02

mpd:03



```
1 prefix :<http://www.noveleer.com#>
2 prefix owl:<http://www.w3.org/2002/07/owl#>
3 prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>
5
6 SELECT *
7
8 //ค้นว่า :nv002 type เป็นอะไร
9 WHERE {
10   :nv002 rdf:type ?type.
11 }
12
13 //ค้นว่าคลาส Novel มีรหัสบังอະໄຣ และมีtitle และtagline อະໄຣນັ້ງ
14 WHERE {
15   ?novel a :Novel ;
16   :title ?title ;
17   :tagline ?tagline .
18 }
19
20 //ค้นว่า Novel :nv001 มีนิยามก่อนหน้า(hasPreviousNovel) ชื่อเรื่องອະໄໄສTransitive
21 WHERE {
22   :nv001 :hasPreviousNovel ?Novel .
23 }
24
25 //ค้นว่าคลาส ?Author equivalentClass กับคลาสອະໄໄສ
26 SELECT *
27 WHERE {
28   ?Author owl:equivalentClass ?Class .
29 }
30
31 //ค้นว่า :rvvmr001 มี rating เท่าไหร
32 WHERE {
33   :rvvmr001 :rating ?rating .
34 }
35
36 //ค้นว่า writer :wr001 sameAs กับ writer อະໄຣນັ້ງ
37 WHERE {
38   :wr001 owl:sameAs ?writer.
39 }
40
41 //ค้นว่า reviewer:rvw001 review อະໄຣນັ້ງ
42 WHERE {
43   :rvw001 :review ?allreview.
44 }
45
46 invertOf ( //ค้นว่า นิยามรหัสนີ້ รีວิวโดย(reviewBy) reviewer คนໃຫນ
47 WHERE {
48   :rvdd001 :reviewBy ?Reviewer .
49 }
50
51 //ค้นว่า นิยามรหัสນີ້ รีວิวโดย(reviewBy) reviewer คนໃຫນ
52 WHERE {
53   :rvw003 :review ?Review .
54 } )
55
56 //ค้นว่า Novel มี subClassOf อະໄຣນັ້ງ
57 WHERE {
58   ?Class rdfs:subClassOf :Novel .
59 }
```