**Semantic Web**

2008.07.17 [ **조대웅, 한광희** ]

**환경**

Java, Tomcat, Servlet, Jena-2.5.6

**File Description**

Test.html // OntologyMap 기본 인터페이스

Jenatest.java // Classes 서블릿

Jenatest2.java // Property 서블릿

Jenatest3.java // Individuals 서블릿

Jenatest4.java // MyungHoKim 서블릿

Jenatest5.java // Books 서블릿

Jenatest6.java // Students 서블릿

seClass.java //클래스 보여주는 클래스

Pro.java //프로퍼티 보여주는 클래스

Individuals.java // Individuals 보여주는 클래스

Profe.java // 교수님 정보 보여주는 클래스

Book.java // 책 보여주는 클래스

Stud.java // 학생 보여주는 클래스

예제가 잘 나와있지 않으므로, “Jena API” 를 잘 찾아 쓰는 것이 중요하다.

[ D:\Jena-2.5.6\doc\javadoc\index.html ]

**Jena**는 HP 시맨틱 웹 연구소(HP Labs Semantic Web Research)의 Brian McBride에 의해 개발된 시맨틱 웹 프레임워크이다. RDF, RDFS 및 OWL을 위한 프로그래밍 환경과 기본적인 RDF파서를 제공하며 내부적으로 룰(rule)기반의 추론엔진을 포함하고 있다.Jena 프레임워크는 다음을 포함한다.

-RDF API

-RDF를 읽고 RDF/XML, N3, N-Triles로 출력

-OWL API

-기억 및 지속적 저장

-온톨로지 언어를 위한 질의어 SPRQL

**[ Jena RDF Tutorial ]**

**Resource 와 Property**

- Resources have properties. Each property has a value. In this case the value is a literal, which for now we can think of as a strings of characters.

- Resource는 Subject라고 생각하면 된다. Property는 predicate와 value로 이루어져 있다.

**Interface Model**

RDF Model은 Statements의 집합이다. Methods는 다음과 같은 목적으로 제공된다.

- resources, properties, literals 그리고 그들을 연결하는 Statements를 만들기 위해

- model로부터 statements를 추가하고 제거하거나 querying하기 위해

- model들을 결합시키는 연산들의 집합

Models는 Resources[URI nodes and bnodes]를 만들 것이다. Resource를 생성하는 것은 Resource를 model안에서 visible하게 만드는 것이 아니다. 만약 Statements가 resources에 관해서 Model에 추가된다면 Resource는 오직 Models 안에 있다. 유사하게 Model로부터 Resource를 제거하는 유일한 방법은 그것을 언급한 Statements를 제거하는 것이다.

Resource와 Literal이 Model에 의해 생성 됬을 때 Model은 존재하는 올바른 values를 갖는 Resource와 Literal object에 대해서는 자유롭게 재사용할 수 있으며 새로운 하나를 생성하는 것도 가능하다. 모든 Jena RDFNodes(Resource와 Literal의 부모 Interface)와 Statements는 immutable하고 이것은 일반적으로 안전하다.

**[Example code ]**

**import** com.hp.hpl.jena.rdf.model.\*;

**import** com.hp.hpl.jena.util.FileManager;

**import** com.hp.hpl.jena.vocabulary.\*;

**import** java.io.\*;

/\* Tutorial 7 - selecting the VCARD resources \*/

**public** **class** Tutorial07 **extends** Object {

**static** **final** String *inputFileName* = "vc-db-1.rdf"

**public** **static** **void** main (String args[]) {

// create an empty model

Model model = ModelFactory.*createDefaultModel*();

// use the FileManager to find the input file

InputStream in = FileManager.*get*().open(*inputFileName*);

**if** (in == **null**) {

**throw** **new** IllegalArgumentException( "File: " + *inputFileName* + " not found");

}

// read the RDF/XML file

model.read( in, "");

// select all the resources with a VCARD.FN property

ResIterator iter = model.listSubjectsWithProperty(VCARD.*FN*);

**if** (iter.hasNext()) {

System.*out*.println("The database contains vcards for:");

**while** (iter.hasNext()) {

System.*out*.println(" " + iter.nextResource()

.getRequiredProperty(VCARD.*FN*)

.getString() );

}

} **else** {

System.*out*.println("No vcards were found in the database");

}

}

}

**\*. jena 설치**

- jena를 다운 받는다.

- jena에서 제공하는 package들을 사용할 수 있도록 한다.

: jena의 lib 폴더내 \*.jar 파일들을 java jdk 폴더내 jre/lib/ext 폴더에 복사한다.

: 이렇게만 하면, jena 패키지를 코드에서 import 해서 사용할 수 있다.

**\*. jena에 있는 class와 method들.**

**- ModelFactory class**

: Model class의 instance 생성과 관련된 method들을 가진다.

: jena에서 model은 RDF 형식의 그래프를 의미한다.

그러므로, model은 그래프를 표현하는 node, edge 정보를 포함한다.

**- ModelFactory.createDefaultModel**

: 비어 있으면서, 메모리에 올라와 있는(in-memory) model을 하나 생성한다.

**- Model.createResource**

: model에서 하나의 resource를 만든다.

RDF에서 resource는 graph에서 하나의 node를 구성한다.

**- Model.createProperty**

: model에서 하나의 property를 만든다.

RDF에서 property는 graph에서 edge label 이다.

**- Model.createStatement**

: model에서 하나의 statement를 만든다.

RDF에서 statement는 subject - predicate - object 를 의미한다. (predicate = property)

: createStatement에 의해서 statement가 만들어지지만, model에서 추가되지 않는다.

만들어진 statement를 model에 추가하기 위해서는 model.add 를 사용한다.

**- Resource.addProperty**

: resource에 property를 추가한다.

**OWL**

* OWL (Lite, DL, Full) 갈수록 표현력이 향상되는 대신에 효율적인 추론 기능의 지원이 여려워진다.
* OWL 은 추가적인 어휘를 통해 RDF(S) 의 표현력을 확장시켜준다.

**OWL 저장 관리자는**

온톨로지 데이터를 파싱하고 의미별로 재구성하여 저장함.

온톨로지 데이터를 주어, 서술어, 목적어의 트리플들로 재구성하여 관계형 데이터 베이스 관리 시스템에 저장한다.

온톨로지 데이터에서 다른 중요한 정보는 크래스와 프로퍼티들의 계층 정보이다.

**OWL-QL 질의 처리기는**

OWL-QL 질의를 받아서 사용자가 원하는 정보를 추출한다.

**클래스, 개체 및 클래스 계층구조**

* Extension(외연), intension(내연) => 두 클래스의 인터스턴스가 모두 같은지? 의미도 같은지?
* 클래스, 인스턴스
* <owl:Class rdf:ID="레이저 프린터">

<rdfs:subClassOf rdf:resource="#프린터"/>

</owl:Class>

**속성과 계층 구조**

* ObjectProperty - 클래스의 인스턴스를 다른 클래스에 속한 인스턴스와 연결하는 속성.
* DatatypeProperty - 클래스의 인스턴스를 특정한 데이터 타입과 연결하는 속성.
* OWL에서 ObjectProperty 에 해당하는 속성을 정의할 때에는 rdfs : domain 과 rdfs : range 라는 어휘를 사용하여 정의역(domain) 과 공역(range) 을 지정할 수 있다. RDF Schema 에서 언급했던 것과 같다
* <owl : ObjectProperty rdf : ID="제조하다">

<rdfs : domain rdf : resource="#프린터"/>

<rdfs : range rdf : resource="#제조회사"/>

</owl:ObjectProperty>

* 속성도 클래스와 마찬가리조 계층을 형성할 수 있다.

**[OWL의 새로운 기능]**

**클래스의 불(Boolean) 연산[DL]**

* owl : intersectionOf , owl : unionOf, owl : complementOf
* <owl : Class rdf : ID="사람">

<owl : unionOf>

<owl : Class rdf : ID ="#여자"/>

<owl : Class rdf : ID ="#남자">

<owl : complementOf rdf: resource="#여자"/>

</owl : Class>

</owl : unionOf>

</owl : Class>

|  |  |
| --- | --- |
| **열거형 클래스(Enumerated Class)** | **[one of, Thing]** |

<owl:Class rdf:ID="WineColor">

<rdfs:subClassOf rdf:resource="#WineDescriptor"/>

<owl:oneOf> rdf:parseType="Collection">

<owl : Thing rdf :about="#White" />

<owl : Thing rdf :about="#Rose" />

<owl : Thing rdf :about="#Red" />

</owl:oneOf>

</owl:Class>

|  |  |
| --- | --- |
| **클래스의 비접합성[DL]** | **[disjointWith]** |

<owl:Class rdf:ID="레이저 프린트">

<owl : disjointWith rdf:resource="#잉크젯 프린터"/>

</owl:Class>

|  |  |
| --- | --- |
| **RDF** | **Resource Description Framework** |

* 단순한 트리플(triple) 형태로 웹 자원을 기술하는 언어이다.
* 자원(주어)-속성(서술부)-속성값(목적부)
* URI 는 웹 상에 존재하는 자원을 지칭하는 스트링의 표준 형식

|  |  |
| --- | --- |
| URL | URN |
| 실제 네트워크 경로를 나타냄 | 임의의 자원을 가리키는 영속적이고 고유한 이름으로, 자원이 저장되어 있는 위치와는 무관 |
|  | <Http://on.sun.ac.kr/book/indexl.html#section>  #은 URI와 단편 식별자를 분리 해주는 역할.  단편 식별자인 section 은 네임 스페이스에 해당하는 앞의 URI 안에서 효력이 있는  자원의 식별자라고 할 수 있다. |

**RDF Schema**

* RDF 에는 속성의 도메인을 제한하거나 비슷한 자원을 한 데 묶어서 클래스로 표현하는 기능이 없다.