PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
MAESTRÍA EN INFORMÁTICA

INGENIERÍA Y GESTIÓN DEL CONOCIMIENTO
Laboratorio 1 - RDF
(Segundo Semestre del 2020)

Objetivos de aprendizaje:

- Configurar un proyecto en Netbeans para manipular RDF.
- Crear y salvar modelos RDF en archivos.
- Leer y consultar modelos RDF a partir de archivos.

---

1. Introducción

En este laboratorio se busca que el alumno manipule estructuras RDF para resolver problemas de la Ingeniería del Conocimiento. Se busca que el alumno pueda crear, salvar, leer y consultar modelos RDF usando un lenguaje de programación, aplicando de esta manera los conceptos vistos en la sesión teórica.

2. Métodos y procedimientos

Se utilizarán las siguientes herramientas:

- Como lenguaje de programación se utilizará Java.
- Como Entorno de Desarrollo Integrado (IDE) se utilizará Netbeans 8.2 el cual puede ser descargado a partir del URL `https://netbeans.org/downloads/8.2/`.
- Para la manipulación del RDF se utilizará la API de Apache Jena, el cual se encuentra en el URL `https://jena.apache.org/`. El alumno podrá encontrar un tutorial de RDF usando Jena en el URL `https://jena.apache.org/tutorials/rdf_api.html`.

Parte I
Proyecto Apache Jena

3. Creación de un proyecto en Netbeans

En esta sección se procederá a crear un proyecto en Netbeans. NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.

3.1. Creación de un nuevo proyecto

Para crear un proyecto, se debe utilizar la opción del menú `File`, luego la opción `New Project...`. Aparecerá una pantalla como la que se presenta en la figura 1. En esta pantalla se debe seleccionar la opción `Java` dentro de la lista de categorías y `Java Application` dentro de la lista de tipos de proyectos. Posteriormente, se debe seleccionar el botón `Next >`. Aparecerá una pantalla en donde se deberá configurar los datos del proyecto conforme a la figura 2.

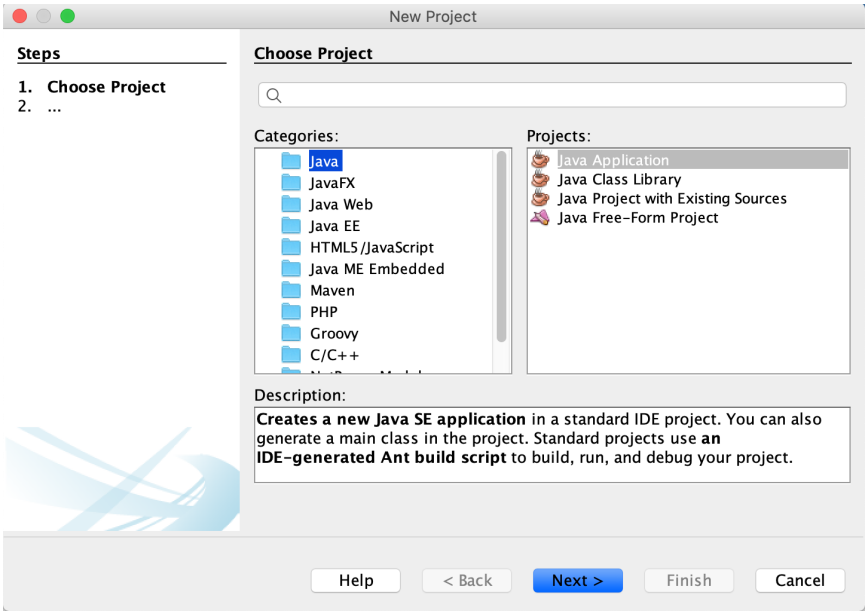

Figura 1: Netbeans: selección del tipo de proyecto.

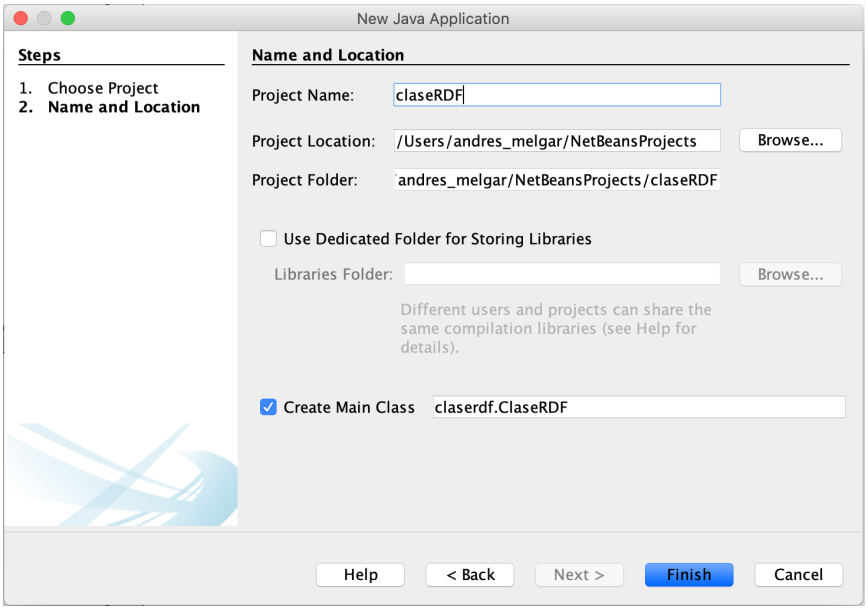

Figura 2: Netbeans: configurando los datos del proyecto.

3.2. Configurando los datos del proyecto

Se debe indicar el `Project name` que será el nombre que se usará para crear la carpeta que contendrá todos los archivos del proyecto. En el `Project location` deberá indicar la ruta en donde desea que se cree la carpeta del proyecto.

Luego de presionar el botón `Finish`, el proyecto aparecerá en el panel de proyecto que se encuentra a la izquierda del IDE tal como se puede apreciar en la figura 3.

3.3. Incluyendo el paquete de soporte a Jena

Para incluir el soporte a Jena en la aplicación, descargaremos la última distribución de Jena. Esta distribución se encuentra en el URL `https://downloads.apache.org/jena/binaries/apache-jena-3.16.0.zip`. Luego de descargar la distribución de Jena, proceda a descomprimir el archivo, los archivos `jar` que se incorporarán en el proyectos serán las que se encuentran en la carpeta `lib`.

Para incluir los `jar`, en el panel de proyectos, deberá seleccionar la carpeta `Libraries`, luego seleccio-

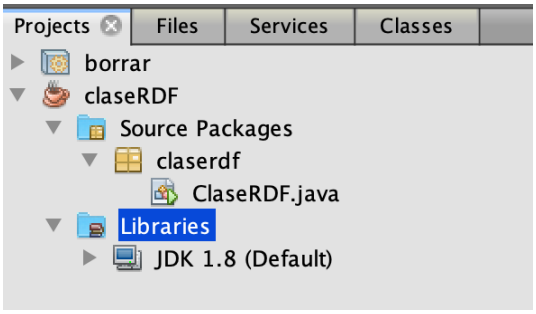

Figura 3: Netbeans: visualizando el proyecto en el panel.

namos el menú secundario y escogemos la opción `Add JAR/Folder...` tal como se muestra en la figura 4.

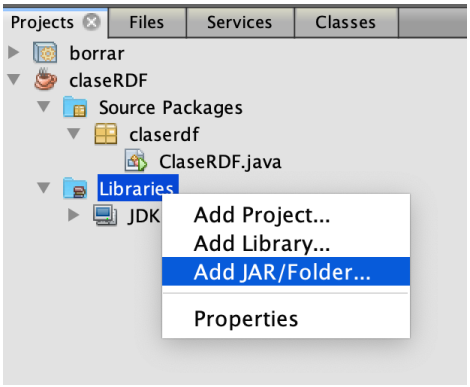

Figura 4: Netbeans: incluyendo los jar.

Luego de seleccionar esta opción se presentará una pantalla para seleccionar el folder en donde se encuentran los archivos `jar`. Esta pantalla será similar a la presentada en la figura 5.

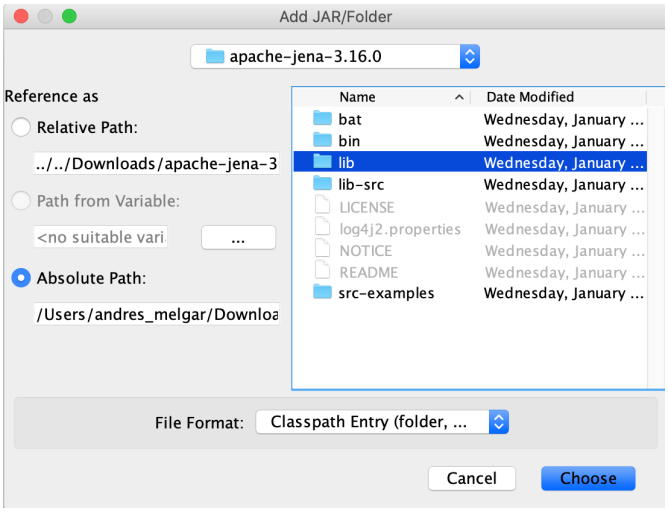

Figura 5: Netbeans: seleccionando los jar.

Parte II
Creando modelos RDF

4. La Ontología vCard

En esta parte se crearán esquemas RDF usando la API de Jena en Java. Para esto usaremos la ontología vCard. vCard es un formato estándar para el intercambio de información personal, específicamente tarjetas personales electrónicas[1]. Existe una ontología denominada VCard que implementa la especificación RFC6350 en RDF. Dicha ontología se puede encontrar en el URL `https://www.w3.org/TR/vcard-rdf/`.

5. Creación de modelos RDF

Lo primero a realizar es la creación de una clase en Java para colocar el código para la creación del modelo RDF. Para esto se deberá seleccionar la carpeta `Source Packages`, luego seleccionamos el menú secundario y escogemos la opción `New` y seleccionamos el submenú `Java Class...` tal como se puede apreciar en la figura 6.

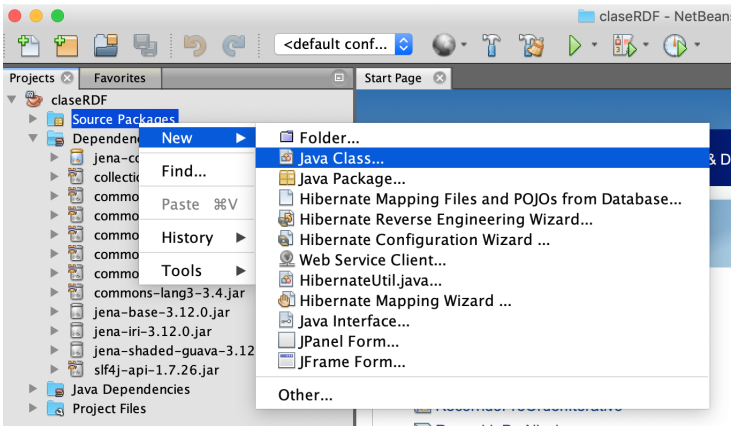

Figura 6: Netbeans: incluyendo clase de Java.

Una vez realizado esto, se presentará un formulario como el de la figura 7 en donde se deberá especificar el nombre del clase en el campo `Class Name`, aunque no es obligatorio, conviene que seleccione el paquete en donde se colocará la clase. Para este fin se utiliza el campo `Package`.

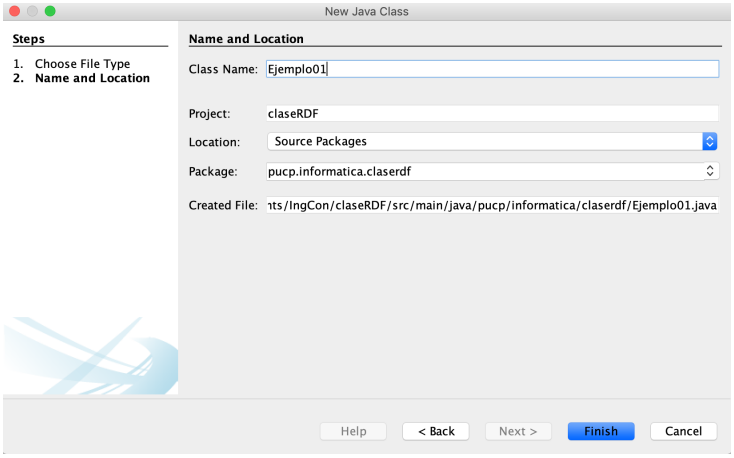

Figura 7: Netbeans: configurando clase de Java.

[1] Mayor información puede encontrarla en el URL `https://es.wikipedia.org/wiki/VCard`.

Como ya se ha visto en la sesión de clase teórica, una tripleta RDF está formada por un sujeto, un predicado y un objeto. Al sujeto, que es el recurso de la Web Semántica, se le va a representar mediante un URI[2]. Para este ejemplo se usará el URI `http://www.pucp.edu.pe/AndresMelgar`, pero usted podrá usar el URI que considere más conveniente. El objeto para la tripleta ejemplo será `Andrés Melgar`, una cadena de caracteres. El predicado para el ejemplo será `http://www.w3.org/2001/vcard-rdf/3.0#FN` que es una propiedad definida en la ontología vCard. La propiedad `http://www.w3.org/2001/vcard-rdf/3.0#FN` o en corto `FN` permite asignar el nombre completo de un recurso. Este campo es el único que es obligatorio para los vCard.

En la figura 8 se puede apreciar la tripleta RDF presentada en este ejemplo de forma gráfica. Esta imagen ha sido generada a través del servicio de validación de la W3C el cual se puede encontrar en el URL `https://www.w3.org/RDF/Validator/`.

Figura 8: RDF: tripleta "Andrés Melgar".

Para manipular RDFs usando el API de Jena, se debe crear una instancia de la clase `Model`. En el programa 1 se puede apreciar cómo se realiza la instanciación del modelo en la línea 15. El modelo posee el método `model.createResource` que como su nombre lo indica, permite crear un recurso pasándole como parámetro su correspondiente URI. La vinculación de sujeto con la propiedad y con el objeto se hace a través del método `addProperty` que se encuentra definida en la jerarquía de la clase `Resource`. En la línea 17 podrá ver esta vinculación.

Programa 1: Creando RDF en Jena

```
1  package pucp.informatica.claserdf;
2
3  import org.apache.jena.rdf.model.Model;
4  import org.apache.jena.rdf.model.ModelFactory;
5  import org.apache.jena.rdf.model.Resource;
6  import org.apache.jena.vocabulary.VCARD;
7
8  public class Ejemplo01 {
9
10     public static void main(String[] args) {
11         System.out.println("Creando RDF...");
12         String personURI = "http://www.pucp.edu.pe/AndresMelgar";
13         String fullName = "Andrés Melgar";
14
15         Model model = ModelFactory.createDefaultModel();
16         Resource andresMelgar = model.createResource(personURI);
17         andresMelgar.addProperty(VCARD.FN, fullName);
18     }
19 }
```

6. Serialización de modelos RDF

Para serializar un modelo RDF usando el API de Jena, se usa el método `write` definido para la clase `Model`. El método `write` se encuentra sobrecargado, pero una de las versiones más usada utiliza dos parámetros. El primero para especificar el flujo de salida y el segundo para especificar la notación de la salida.

El segundo parámetro puede ser:

- `RDF/XML` Para obtener el modelo usando notación RDF/XML en formato completo.

[2]Recuerde que el URI es un identificador de recursos uniforme o URI, proviene del inglés *Uniform Resource Identifier*.

- `RDF/XML-ABBREV` Para obtener el modelo usando notación RDF/XML en formato abreviado.
- `N-TRIPLE` Para obtener el modelo usando notación N-Triple. Más información la puede encontrar en el URL `https://www.w3.org/TR/n-triples/`.
- `TURTLE` Para obtener el modelo usando notación Turtle. Más información la puede encontrar en el URL `https://www.w3.org/TR/turtle/`.
- `TTL` Para obtener el modelo usando notación Turtle. Más información la puede encontrar en el URL `https://www.w3.org/TR/turtle/`.
- `N3` Para obtener el modelo usando notación Notacion3. Más información la puede encontrar en el URL `https://www.w3.org/TeamSubmission/n3/`.

En el programa 2 se puede apreciar un ejemplo de código en Java que invoca a dicho método. Como es de esperarse, este código debe usar luego de haber leído o construido la estructura RDF.

Programa 2: Serializando un RDF en Jena

```
1         ...
2         System.out.println("Write RDF RDF/XML...");
3         model.write(System.out, "RDF/XML");
4         ...
```

La salida obtenida luego de incluir el código descrito en el programa 2 en el programa 1 es el siguiente.

```
Write RDF RDF/XML...
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#" >
  <rdf:Description rdf:about="http://www.pucp.edu.pe/AndresMelgar">
    <vcard:FN>Andrés Melgar</vcard:FN>
  </rdf:Description>
</rdf:RDF>
```

Para poner en práctica

- Altere el programa 1 para serializar el modelo RDF creado. Use el código presentado en el programa 2.
- Prueba con todas las opciones de notación de salida en la serialización del modelo RDF y analice cuáles son las diferencias entre ellas.
- Altere el programa 1 para incluir a más personas. Use las personas que se encuentran en su grupo de trabajo. Cree un método en la clase en Java que permita generalizar la creación de personas usando la ontología vCard.
- Serialice el modelo RDF de la última alteración usando la notación RDF/XML y visualice el grafo RDF usando el validador de la W3C que se encuentra en el URL `https://www.w3.org/RDF/Validator/`.

7. Salvando de modelos RDF en archivos

Para salvar el modelo RDF a un archivo, se utiliza nuevamente el método `write` definido para la clase `Model`. La única diferencia con el ejemplo anterior es que el flujo de salida se dirige hacia un archivo. En el programa 3 se presenta una sección de código que graba el modelo generado en un archivo cuyo nombre es `vCard.rdf`. Note que para usar este segmento de código, se deben importar las clases `FileNotFoundException` y `FileOutputStream`.

Programa 3: Salvando un modelo RDF en Jena

```java
package pucp.informatica.claserdf;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;

        ...
        FileOutputStream output = null;
        try {
            output = new FileOutputStream("vCard.rdf");
        } catch (FileNotFoundException e) {
            System.out.println("Ocurrió un error al crear el archivo.");
        }
        model.write(output, "RDF/XML-ABBREV");
        ...
```

Para poner en práctica

- Altere el programa 1 para salvar el modelo RDF creado en un archivo. Use el código presentado en el programa 3.
- Genere un archivo `vCard.rdf` que represente el resultado del modelo `RDF/XML-ABBREV`.
- Genere un archivo `vCard.nt` que represente el resultado del modelo `N-TRIPLE`.
- Genere un archivo `vCard.ttl` que represente el resultado del modelo `TURTLE`.
- Compare los 3 archivos y analice las diferencias entre ellos.

8. Manipulando nodos en blanco

Como ya se comentó en la sesión teórica, los nodos en blanco son nodos en un grafo RDF que no poseen URI, ya sea porque el autor del documento no sabe, no conoce, no quiere o no necesita proveer un nombre. En la figura 9 se puede apreciar un grafo RDF con un nodo en blanco. En dicha figura el nodo en blanco aparece etiquetado con `genid:UA0`. En este ejemplo fue necesario la utilización de un nodo en blanco pues a la propiedad `N` (Name) se le puede vincular con nombres estructurados como por ejemplo `Family Name` o `Given Name`.

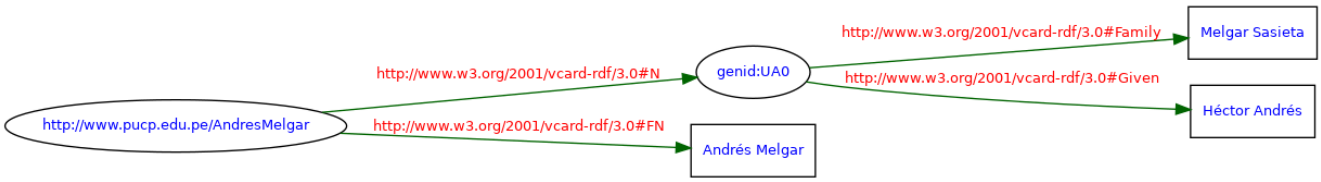

Figura 9: RDF: tripleta “Andrés Melgar” con nodo en blanco.

En el programa 4 se puede apreciar la creación de un nodo en blanco en la línea 21. Note como en este caso no se pasa el URI como parámetro al método `createResource`.

Programa 4: Creando un nodo en blanco en Jena

```java
package pucp.informatica.claserdf;

import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.Resource;
import org.apache.jena.vocabulary.VCARD;
```

```java

public class Ejemplo02 {

    public static void main(String[] args) {
        System.out.println("Creando RDF...");
        String personURI = "http://www.pucp.edu.pe/AndresMelgar";
        String fullName = "Andrés Melgar";
        String givenName = "Héctor Andrés";
        String familyName = "Melgar Sasieta";

        Model model = ModelFactory.createDefaultModel();
        Resource andresMelgar = model.createResource(personURI);
        andresMelgar.addProperty(VCARD.FN, fullName);

        Resource blankNode = model.createResource();
        blankNode.addProperty(VCARD.Given, givenName);
        blankNode.addProperty(VCARD.Family, familyName);
        andresMelgar.addProperty(VCARD.N, blankNode);

        System.out.println("Write RDF RDF/XML...");
        model.write(System.out, "RDF/XML");
    }
}
```

El programa anterior serializa el modelo de la siguiente manera:

```
Write RDF RDF/XML...
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#" >
  <rdf:Description rdf:about="http://www.pucp.edu.pe/AndresMelgar">
    <vcard:N rdf:nodeID="A0"/>
    <vcard:FN>Andrés Melgar</vcard:FN>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <vcard:Family>Melgar Sasieta</vcard:Family>
    <vcard:Given>Héctor Andrés</vcard:Given>
  </rdf:Description>
</rdf:RDF>
```

Para poner en práctica

- Altere el programa 4 para incluir a más personas y represente sus nombres (`Given`), apellidos (`Family`) y nombre completo (`FN`). Use las personas que se encuentran en su grupo de trabajo. Cree un método en la clase en Java que permita generalizar la creación de personas usando la ontología vCard.
- Para cada persona incluida ahora registre su correo electrónico. Use la propiedad `EMAIL` la cuya URI se puede obtener en Jena con la instrucción `VCARD.EMAIL`. Esta propiedad puede asignarle directamente al recurso. Si la persona posee más de una dirección de correo electrónica, registre todas ellas.
- Para cada persona incluida ahora registre su teléfono. Use la propiedad `TEL` la cuya URI se puede obtener en Jena con la instrucción `VCARD.TEL`. Esta propiedad puede asignarle directamente al recurso. Si la persona posee más de un teléfono, registre todos ellos.
- Salve el modelo RDF creado en un archivo.
  - Genere un archivo `vCard.rdf` que represente el resultado del modelo `RDF/XML-ABBREV`.
  - Genere un archivo `vCard.nt` que represente el resultado del modelo `N-TRIPLE`.
  - Genere un archivo `vCard.ttl` que represente el resultado del modelo `TURTLE`.

La salida el programa deberá ser similar a:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#" >
  <rdf:Description rdf:about="http://www.pucp.edu.pe/AndresMelgar">
    <vcard:N rdf:nodeID="A0"/>
    <vcard:TEL>987654321</vcard:TEL>
    <vcard:TEL>123456789</vcard:TEL>
    <vcard:EMAIL>amelgar@gmail.com</vcard:EMAIL>
    <vcard:EMAIL>amelgar@pucp.edu.pe</vcard:EMAIL>
    <vcard:FN>Andrés Melgar</vcard:FN>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <vcard:Family>Melgar Sasieta</vcard:Family>
    <vcard:Given>Héctor Andrés</vcard:Given>
  </rdf:Description>
</rdf:RDF>
```

Parte III
Leyendo modelos RDF

9. Lectura de un modelo en Jena

El archivo `vCard`, que se encuentra en la plataforma Paideia, representa el grafo RDF que se presenta en la figura 10. Este archivo ha sido salvado en las formatos `.rdf`, `.ttl` y `.nt`.

Leer de un archivo usando la API de Jena es una tarea relativamente simple. En el programa 5 se puede apreciar un programa que lee un modelo RDF del archivo `vCard.rdf`. Se continua usando la clase `Model` para almacenar el modelo con la única diferencia que ahora se carga este de un archivo tal como se puede apreciar en la línea 13.

Figura 10: Grafo de vCard.

Programa 5: Leyendo un modelo RDF de un archivo

```
1 package pucp.informatica.claserdf;
```

```java

import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;

public class Ejemplo03 {

    public static void main(String[] args) {
        System.out.println("Leyendo RDF...");
        Model model = ModelFactory.createDefaultModel();
        String inputFileName = "vCard.rdf";

        model.read(inputFileName);
        model.write(System.out);
    }
}
```

Para poner en práctica

- Verifique que se pueda leer el archivo `vCard.rdf` en un modelo RDF.
- Verifique que se pueda leer el archivo `vCard.nt` en un modelo RDF.
- Verifique que se pueda leer el archivo `vCard.ttl` en un modelo RDF.

Parte IV
Consultando modelos RDF

10. Consultando todas las tripletas del modelo RDF

La API de Jena dispone de un método que permite consultar las tripletas de un modelo RDF. Este método es `listStatements` y se encuentra disponible en la clase `Model`. Este método retorna un iterador de tripletas. Para poder recuperar la tripleta se usa la clase `Statement`. Luego cada elemento de la tripleta se puede descomponer en las clases `Resource`, `Property` y `RDFNode`. La clase `Resource` representa el sujeto, la clase `Property` representa al predicado y la clase `RDFNode` representa al objeto. En el programa 6 se puede apreciar cómo se consultan e imprimen todas las tripletas de un modelo RDF.

Programa 6: Consultando todas las tripletas del modelo RDF

```java
package pucp.informatica.claserdf;

import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.Property;
import org.apache.jena.rdf.model.RDFNode;
import org.apache.jena.rdf.model.Resource;
import org.apache.jena.rdf.model.Statement;
import org.apache.jena.rdf.model.StmtIterator;

public class Ejemplo04 {

    public static void main(String[] args) {
        String inputFileName = "vCard.rdf";
        Model model = ModelFactory.createDefaultModel();
        model.read(inputFileName);

        StmtIterator iter = model.listStatements();
        while (iter.hasNext()) {
            Statement stmt = iter.nextStatement();
```

```
            Resource subject = stmt.getSubject();
            Property predicate = stmt.getPredicate();
            RDFNode object = stmt.getObject();

            System.out.print(subject.toString());
            System.out.print(" " + predicate.toString() + " ");
            if (object instanceof Resource) {
                System.out.print(object.toString());
            } else {
                System.out.print("\"" + object.toString() + "\"");
            }
            System.out.println(".");
        }
    }
}
```

El resultado del programa 6 es similar a lo siguiente:

```
a62b90d6-3465-49fe-b8db-2383293d7ee5 http://www.w3.org/2001/vcard-rdf/3.0#Given "Mendoza".
a62b90d6-3465-49fe-b8db-2383293d7ee5 http://www.w3.org/2001/vcard-rdf/3.0#Family "Andrés Mendoza".
http://www.pucp.edu.pe/RobertoMelgar http://www.w3.org/2001/vcard-rdf/3.0#FN "Roberto Melgar".
http://www.pucp.edu.pe/RobertoMelgar http://www.w3.org/2001/vcard-rdf/3.0#EMAIL "rmelgar@pucp.edu.pe".
...
...
```

Para poner en práctica

Tomando como base el resultado de la impresión de las tripletas:

- Identifique el sujeto, predicado y objeto de cada tripleta.
- Identifique los nodos en blanco e interprételos.

11. Consultando todos los sujetos del modelo RDF

La API de Jena dispone de un método que permite consultar todos los sujetos de un modelo RDF. Este método es `listSubjects` y se encuentra disponible en la clase `Model`. Este método retorna un iterador de recursos. En el programa 7 se puede apreciar cómo se consultan e imprimen todas los sujetos de un modelo RDF.

Programa 7: Consultando todos los sujetos del modelo RDF

```
package pucp.informatica.claserdf;

import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.ResIterator;
import org.apache.jena.rdf.model.Resource;

public class Ejemplo05 {

    public static void main(String[] args) {
        String inputFileName = "vCard.rdf";
        Model model = ModelFactory.createDefaultModel();
        model.read(inputFileName);

        ResIterator resIter = model.listSubjects();
        while (resIter.hasNext()) {
            Resource res = resIter.nextResource();
            System.out.println(res);
```

```
        }
    }
}
```

12. Consultando todos los sujetos con determinada propiedad del modelo RDF

La API de Jena dispone de un método que permite consultar todos los sujetos con determinada propiedad de un modelo RDF. Este método es `listSubjectsWithProperty` y se encuentra disponible en la clase `Model`. Este método retorna un iterador de recursos. En el programa 8 se puede apreciar cómo se consultan e imprimen todas los sujetos que presentan la propiedad `FN` de un modelo RDF.

Programa 8: Consultando todos los sujetos con determinada propiedad del modelo RDF

```
package pucp.informatica.claserdf;

import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.ResIterator;
import org.apache.jena.rdf.model.Resource;
import org.apache.jena.vocabulary.VCARD;

public class Ejemplo06 {

    public static void main(String[] args) {
        String inputFileName = "vCard.rdf";
        Model model = ModelFactory.createDefaultModel();
        model.read(inputFileName);

        ResIterator resIter = model.listSubjectsWithProperty(VCARD.FN);
        while (resIter.hasNext()) {
            Resource res = resIter.nextResource();
            System.out.print(res);
            String fullName = res.getProperty(VCARD.FN).getString();
            System.out.println(" " + fullName);
        }

    }
}
```

13. Consultando tripletas del modelo RDF usando un selector

La API de Jena dispone de un método que permite consultar tripletas específicas de modelo RDF usando un selector. Este método es `listStatements` y se encuentra disponible en la clase `Model`. Para que se use el selector, este debe ser creado previamente. El selector contiene el patrón de las tripletas que se quiere buscar y se basa en la clase `Selector`. En el programa 9 se puede apreciar cómo se consultan patrones de tripletas específicos e se imprimen todas los tripletas que cumplen dicha patrón.

Programa 9: Consultando tripletas del modelo RDF usando un selector

```
package pucp.informatica.claserdf;

import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.RDFNode;
import org.apache.jena.rdf.model.Resource;
import org.apache.jena.rdf.model.Selector;
import org.apache.jena.rdf.model.SimpleSelector;
```

```java
import org.apache.jena.rdf.model.StmtIterator;

public class Ejemplo07 {

    public static void main(String[] args) {
        String inputFileName = "vCard.rdf";
        Model model = ModelFactory.createDefaultModel();
        model.read(inputFileName);

        String personURI = "http://www.pucp.edu.pe/AndresMelgar";
        Resource person = model.getResource(personURI);
        Selector selector = new SimpleSelector(person, null, (RDFNode)null);
        StmtIterator iter = model.listStatements(selector);
        while (iter.hasNext()) {
            System.out.println(iter.nextStatement().toString());
        }
    }
}
```

Para poner en práctica

- Del modelo RDF imprima el listado de todas las personas (nombres completos) con sus respectivos correos electrónicos.
- Del modelo RDF imprima el listado de todas las personas (nombres y apellidos) con sus respectivos teléfonos.
- Dado un teléfono, recupere e imprima el correo electrónico de la persona vinculada a dicho teléfono.

Parte V
Consultando modelos RDF en la Web

14. Campos de la Investigación y el Desarrollo OCDE

Los Campos de Investigación y Desarrollo (FORD) constituyen un esquema de distribución del conocimiento, propuesto por la Organización para la Cooperación y el Desarrollo Económicos (OCDE), para clasificar las unidades de ejecución de investigación y desarrollo experimental (I+D) y distribuir sus recursos en función del ámbito de conocimiento en el que se lleva a cabo. El Concytec ha desarrollado una modelo RDF de estos campos de investigación a fin de que puedan ser usado de forma estandarizada por distintas organizaciones. La descripción del modelo en HTML se encuentra en el URL `https://concytec-pe.github.io/vocabularios/ocde_ford.html` y la descripción en el formato SKOS se encuentra en el URL `https://concytec-pe.github.io/vocabularios/ocde_ford.xml`.

15. Cargando el modelo desde la Web

Jena hace que el trabajo con RDF sea independiente del lugar de donde se cargue, ya sea desde la memoria, desde un archivo o desde la Web. En el programa 10 se puede apreciar cómo se puede cargar el modelo RDF definido por el Concytec para los campos OCDE a partir de el `https://concytec-pe.github.io/vocabularios/ocde_ford.xml`.

Programa 10: Cargando el modelo desde la Web

```java
package pucp.informatica.claserdf;

import org.apache.jena.rdf.model.Model;
```

```java
import org.apache.jena.rdf.model.ModelFactory;

public class Ejemplo08 {

    public static void main(String[] args) {
        String url =
           "https://concytec-pe.github.io/vocabularios/ocde_ford.xml";
        Model model = ModelFactory.createDefaultModel();
        model.read(url);
        model.write(System.out);
    }
}
```

Parte VI

Creación de un grafo RDF

Dado el grafo de la figura 11, se desea crear el modelo RDF usando el API de Jena y luego salvar el archivo con el nombre `Shakespeare.rdf`. Para crear una propiedad, usaremos el método `createProperty` que se encuentra en la clase `Model`. El método `createProperty` debe recibir como parámetro, el URI de la propiedad.

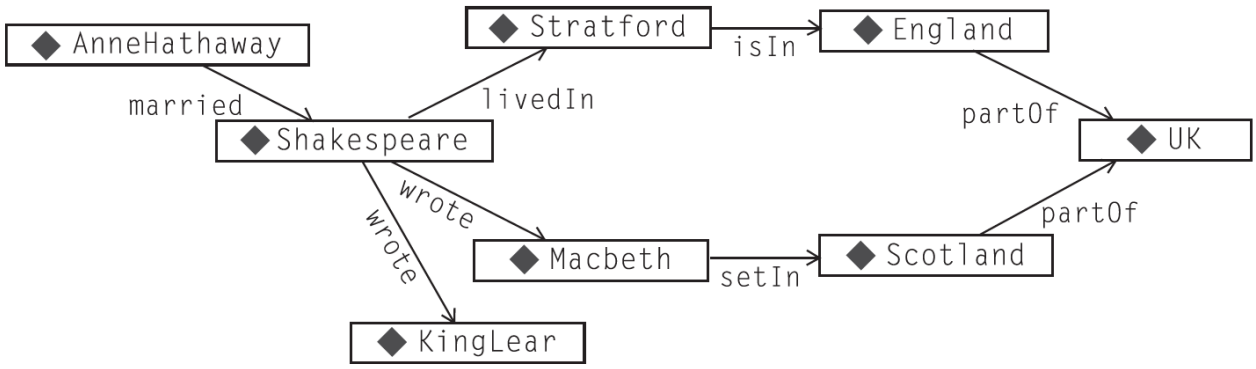

Figura 11: Grafo RDF

Lo primero que haremos será crear el modelo usando la clase `Model` de forma similar a los ejemplos anteriores. Usaremos como URI base a `http://www.pucp.edu.pe/`, la cual identificaremos como `pucp`. Por este motivo, lo setemos como prefijo para el namespace. Usamos para este fin el método `setNsPrefix` de la clase `Model`. En el programa 11 puede ver el uso de este método.

Programa 11: Creando el modelo

```java
        Model model = ModelFactory.createDefaultModel();
        String ns = "pucp";
        String uri="http://www.pucp.edu.pe/";
        model.setNsPrefix(ns, uri);
```

Luego procederemos a crear todas las propiedades del grafo. Para estos usamos el método `createProperty` que se encuentra en la clase `Model`. En el programa 12 puede apreciarse la creación de las propiedades que serán usadas en el grafo RDF.

Programa 12: Creando las propiedades

```java
        Property married = model.createProperty(uri, "married");
```

```
        Property wrote = model.createProperty(uri, "wrote");
        Property liveIn = model.createProperty(uri, "liveIn");
        Property setIn = model.createProperty(uri, "setIn");
        Property isIn = model.createProperty(uri, "isIn");
        Property partOf = model.createProperty(uri, "partOf");
```

Luego procederemos a crear todos los recursos del grafo. Para estos usamos el método `createResource` que se encuentra en la clase `Model`. En el programa 13 puede apreciarse la creación de los recursos que serán usados en el grafo RDF.

Programa 13: Creando los recursos

```
        Resource AnneHathaway = model.createResource("pucp:AnneHathaway");
        Resource Shakespeare = model.createResource("pucp:Shakespeare");
        Resource KingLear = model.createResource("pucp:KingLear");
        Resource Macbeth = model.createResource("pucp:Macbeth");
        Resource Stratford = model.createResource("pucp:Stratford");
        Resource Scotland = model.createResource("pucp:Scotland");
        Resource England = model.createResource("pucp:England");
        Resource UK = model.createResource("pucp:UK");
```

Una vez creadas las propiedades y los recursos, se procede a formar las tripletas. Para estos usamos el método `addProperty` que se encuentra en la clase `Resource`. En el programa 14 puede apreciarse la creación de las tripletas del grafo RDF.

Programa 14: Creando las tripletas

```
        AnneHathaway.addProperty(married, Shakespeare);
        Shakespeare.addProperty(wrote, KingLear);
        Shakespeare.addProperty(wrote, Macbeth);
        Shakespeare.addProperty(liveIn, Stratford);
        Macbeth.addProperty(setIn, Scotland);
        Stratford.addProperty(isIn, England);
        Scotland.addProperty(partOf, UK);
        England.addProperty(partOf, UK);
```

Finalmente, procedemos a salvar el archivo de forma similar a lo realizado en los ejemplos anteriores.

Programa 15: Salvando el RDF

```
        FileOutputStream output = null;
        try {
            output = new FileOutputStream("Shakespeare.rdf");
        } catch (FileNotFoundException e) {
            System.out.println("Ocurrió un error al crear el archivo.");
        }
        model.write(output, "RDF/XML-ABBREV");
```

San Miguel, 9 de septiembre de 2020.