Chapitre 4
Structure documentaire d'une ontologie

1. Introduction

L'ontologie en tant que fichier informatique comporte une structure qui lui est propre. En plus de l'information qu'elle renferme au sujet du domaine du discours, l'ontologie inclut un ensemble de métadonnées qui décrit sa propre nature. Par exemple, son nom, son numéro de version, les ontologies externes auxquelles il fait référence, etc. Le Java OWontologie/L-API contient un ensemble d'API spécialement dévolu à la création, l'impression du contenu, le chargement, la sauvegarde, la manipulation de l'en-tête et du corps du document ontologique.

2. Structure d'un document ontologique

Une ontologie en OWL 2 est un document XML segmenté en trois sections : l'en-tête, le corps et le pied du document.

2.1 En-tête du document ontologique

L'en-tête est la partie du document ontologique qui contient les métadonnées de l'ontologie. L'exemple suivant présente l'en-tête, dans la syntaxe Turtle, de l'ontologie `families` extraite et adaptée du *OWL 2 Web Ontology Language Primer (Second Edition)* (W3C, 2012c).

```
1.  @prefix : <http://example.com/owl/families_ttl/> .
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 xml: <http://www.w3.org/XML/1998/namespace> .
5.  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
6.  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
7. @prefix otherOnt: <http://example.org/otherOntologies/families/> .
8.  @base <http://example.com/owl/families_ttl> .
9.  <http://example.com/owl/families_ttl> rdf:type owl:Ontology ;
10. rdfs:comment "Cette ontologie a été téléchargée du site
    http://www.w3.org/TR/2012/REC-owl2-primer-20121211/"@fr ;
11. owl:versionIRI <http://example.com/owl/families_ttl/1.0.0> ;
12. owl:imports <http://example.org/otherOntologies/families.owl> .
```

Des informations comme l'IRI de base de l'ontologie (ligne 1), la définition des préfixes (lignes 2 à 7), l'association de l'IRI de base à l'ontologie courante (lignes 8 et 9), les commentaires (aussi nommés annotations) associés à l'ontologie (ligne 10), le numéro de version de l'ontologie (ligne 11) ainsi que les IRI des ontologies importées (ligne 12) sont autant d'informations qui figurent dans l'en-tête du document ontologique.

2.2 Corps du document ontologique

Le corps de l'ontologie contient les entités ontologiques (les classes, les propriétés, les individus et les axiomes) qui servent à décrire le domaine du discours. Toujours issu du *OWL 2 Primer*, l'exemple suivant présente un extrait du corps de l'ontologie `families` dans la syntaxe Turtle.

```
. . .
:hasHusband   rdf:type  owl:FunctionalProperty .
:hasHusband   rdf:type  owl:InverseFunctionalProperty .

:hasWife rdf:type           owl:ObjectProperty .
:hasWife rdfs:domain        :Man ;
         rdfs:range         :Woman .
:hasWife rdfs:subPropertyOf :hasSpouse .
```

Cet extrait indique que `:hasHusband` est une propriété fonctionnelle et qu'elle est aussi une propriété fonctionnelle inverse, que `:hasWife` est une propriété d'objets et que `:hasWife` a pour domaine `:Man` et a pour codomaine (`rdfs:range`) `:Woman`, et enfin que `:hasWife` est une sorte de propriété de `:hasParent`.

2.3 Pied du document ontologique

Le pied du document contient la série de métacaractères qui permettent de fermer le document. Dans la syntaxe Turtle et la syntaxe fonctionnelle, il n'existe aucun métacaractère spécifique à la fermeture du document. En revanche, dans la syntaxe XML/RDF la balise `</rdf:RDF>` est employée pour clore le document ontologique.

3. Créer une IRI

Pour créer un document ontologique, la première étape consiste à construire l'IRI de l'ontologie. La méthode statique `create()` de la classe `IRI` permet d'instancier un objet `IRI`. Les exemples de code ci-dessous présentent trois façons distinctes d'instancier l'`IRI` :

– Soit en fournissant une adresse http :

```
IRI pizza_http_iri =
    IRI.create("http://owl.cs.manchester.ac.uk/co-ode-
    files/ontologie/pizza.owl");
```

– Soit en fournissant l'adresse dans le système de gestion de fichiers :

```
IRI pizza_file_iri =
    IRI.create("file:///C:/JavaWebSemantique/workspace/com.java-ws.
    ontologie/WebContent/pizza.owl");
```

– Soit en associant directement un préfixe à l'IRI :

```
IRI pizza_file_full_iri = IRI.create("pizza",
    "http://owl.cs.manchester.ac.uk/co-ode-
    files/ontologie/pizza.owl");
```

Remarque

Attention, la méthode `create()` ne valide pas l'existence de la ressource, elle ne fait que créer l'adresse de la ressource.

4. Gérer le document ontologique

Comme il a été vu, en plus de contenir les entités permettant de décrire un domaine, le document ontologique contient des métadonnées qui décrivent les caractéristiques du document. Dans l'application Java, la classe Java qui gère le document ontologique est l'`OWLOntologyManager`. Elle gère la création, le chargement, la sauvegarde ou la suppression de la partie ontologique du document par la gestion de la classe `OWLOntology`. Les entités de l'ontologie (les individus, les classes, les propriétés, etc.) sont instanciées grâce à la fabrique (*factory*) `OWLDataFactory`. La fabrique est obtenue à partir de la méthode `getOWLDataFactory()` du gestionnaire.

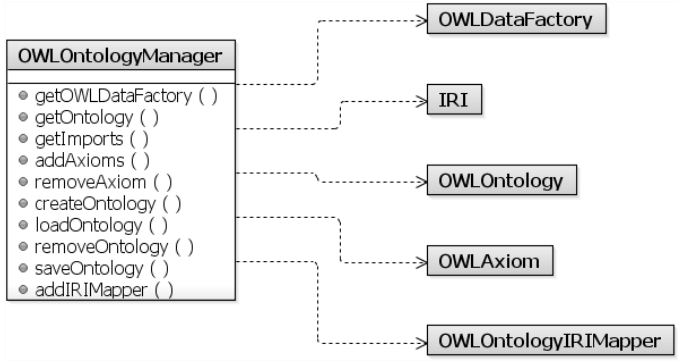

Figure 4.1 : Diagramme UML de la classe `OWLOntologyManager`

```
OWLOntologyManager manager =
    OWLManager.createOWLOntologyManager();
```

Exemple de création d'un `OWLOntologyManager`

5. Instancier une ontologie (OWLOntology)

Il existe deux façons pour instancier une ontologie soit en la créant, soit en la chargeant à partir d'un document déjà existant.

5.1 Créer une ontologie

Le corps du document ontologique est une instance de la classe `OWLOntology`. Dans le document ontologique, la création d'une nouvelle ontologie est assumée par la méthode `createOntology()` du gestionnaire de documents (`manager`). L'`IRI` passée en paramètre à la méthode `createOntology()` définit l'emplacement ainsi que le préfixe de base du document ontologique.

```
OWLOntologyManager manager =
    OWLManager.createOWLOntologyManager();
IRI myFamilyIRI =
    IRI.create("http://java-ws.com/ontologie/myfamily");
OWLOntology myFamilyOntology =
    manager.createOntology(myFamilyIRI);
```

Exemple de création d'un OWLOntology

Dans le résultat de l'exécution de l'exemple, il est constaté que le IRI est assigné au *namespace* (`xmlns`) ainsi qu'à la *base* (`xml:base`) du document ontologique. Les lignes xmlns: subséquentes composent les déclarations nécessaires au document ontologique. La ligne associée à la balise `owl:Ontology` est l'instance de l'ontologie créée lors de l'appel de la méthode `manager.createOntology(myFamilyIRI)`. Finalement, la balise `</rdf:RDF>` indique le pied du document ontologique.

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://java-ws.com/ontologie/myfamily#"
     xml:base="http://java-ws.com/ontologie/myfamily"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology
    rdf:about="http://java-ws.com/ontologie/myfamily"/>
</rdf:RDF>
```

Représentation de l'exemple dans la syntaxe RDF/XML

5.2 Charger une ontologie

Il existe parfois des situations où l'on souhaite charger une ontologie déjà existante. Dans ces cas, il est nécessaire d'employer la méthode `manager.loadOntology()` du gestionnaire de document. Le chargement à partir d'une IRI consiste à indiquer au gestionnaire que le document ontologique est disponible à partir d'une IRI. Dans le cas d'une ressource web, l'IRI indique l'adresse web du document. Dans le cas d'une ressource locale, l'IRI indique l'emplacement du fichier dans le système de gestion de fichiers. La méthode `loadOntology()` traite l'information indépendamment du *scheme* de l'IRI, qu'il soit `http`, `ftp`, `file` ou autre.