第33卷 第14期 计 算 机 工 程 2007年7月
Vol.33 No.14 Computer Engineering July 2007
·软件技术与数据库· 文章编号：1000—3428(2007)14—0059—03 文献标识码：A 中图分类号：TP311

基于 Jena 的本体构建方法研究

向 阳，王 敏，马 强
(同济大学电子信息与工程学院，上海 200092)

摘 要：针对本体构建中构造方法不清晰、本体描述语言不统一、可用工具较少的难题，在 Jena 的基础上提出了基于 Jena 的本体构建方法。该方法由描述类、描述属性、将属性关联到类、定义实例和加入本体维护元数据 5 个步骤组成，有效地解决了本体构建中的难题。最后以一个实例验证了该方法的有效性。
关键词：本体；本体构建；Jena

Research on Jena-based Ontology Building

XIANG Yang, WANG Min, MA Qiang
(School of Electronic Information and Engineering, Tongji University, Shanghai 200092)

【Abstract】There are a lot of difficulties in the ontology building such as the unclear building methods, ununified ontology languages, lack of tools. To solve these problems, this paper presents an ontology building method with Jena. The method is composed of 5 parts: class description, property description, link of property and class, individual creation, ontology metadata adding. The validity of the method is proved with an instance.
【Key words】 ontology; ontology building; Jena

本体是对领域中的概念及概念之间联系的显式描述。具体地说，就是要描述一个领域需要哪些概念，概念由哪些属性标识，属性又具有什么约束，概念对应于哪些实例。

在本体的构建中也存在一些问题：本体构造方法定义不清晰；本体构造语言繁多，不同语言构造出来的本体交互性弱；本体构建工具少，目前可供使用的有斯坦福大学的 Protégé 和 HP 公司的 Jena 等。本文在 Jena 基础上，提出了 OWL 本体构建方法。

1 Jena 体系结构

1.1 Jena 的接口功能

Jena是HP 公司开发的一个基于Java的开放源代码语义网工具包，为解析RDF、RDFS和OWL本体提供了一个编程环境及一个基于规则的推理引擎[1]。语义网标准的核心是作为通用数据结构的RDF图[1]。Jena将RDF图作为其核心的接口。Jena有以下几个主要功能[2]：

(1)RDF API(主要是 com.hp.hpl.jena.rdf.model 包)。可将 RDF 模型视为一组 RDFstatements 集合。

(2)RDQL 查询语言(主要是 com.hp.hpl.jena.rdql 包)。对 RDF 数据的查询语言，可以伴随关系数据库存储一起使用以实现查询优化。

(3)推理子系统(主要是 com.hp.hpl.jena.reasoner 包)。包括基于 RDFS、OWL 等规则集的推理，也可自己建立规则。

(4)内存存储和永久性存储 (主要是 com.hp.hpl.jena.db)。Jena 提供了基于内存暂时存储的 RDF 模型方法，目前仅支持 MySQL、Oracle 和 PostgreSQL 的数据存储。

(5)本体子系统(主要是 com.hp.hpl.jena.ontology 包)。Jena 对 OWL、DAML+OIL 和 RDFS 提供不同的接口支持。

1.2 Jena 的接口结构

Jena 主要由 API，SPI 组成。用户编程只需使用 API。SPI 为 Jena 提供核心数据结构。Jena 库由包来管理，Jena API 以接口方式定义。经常用到包的有：

(1)com.hp.hpl.jena.rdf.model 包，可创建和操纵RDF图，是本体API的基础。结构如图 1 所示[3]。

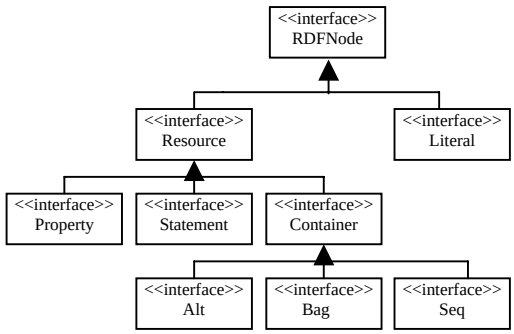

图 1 rdf.model 包主要接口函数

(2)com.hp.hpl.jena.ontology 包。为操纵基于 RDF 的本体提供了抽象接口和实现，结构如图 2 所示。

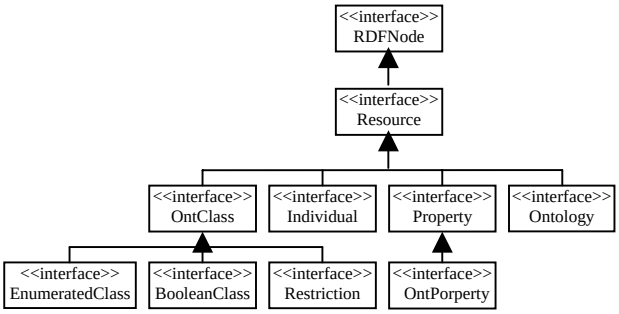

图 2 ontology 包的主要接口函数

基金项目：国家自然科学基金资助项目(70371054)
作者简介：向 阳(1962－)，男，教授、博士生导师，主研方向：语义网，本体，Web 挖掘；王 敏、马 强，硕士研究生
收稿日期：2006-07-25 **E-mail：**drxigyang@rip.sina.com

了解了这几个基本接口之后，就可以使用 Jena 来处理本体了。

2 基于 Jena 的本体构建方法

2.1 方法步骤

本体创建方法没有一个固定的模式，自上而下、自下而上、从中间向两边等创建方法有着不同的优势和缺陷。本文在总结多种本体创建方法的基础上，提出了以 Jena 为工具的循环式本体创建方法，目的是为用户提供一种可参照的符合软件工程的新的本体构建思路。

用户开始定义基于 Jena 的本体构建方法之前，需先定义描述领域中的资源模式图，要尽量多考虑重用性，达到合理描述领域资源的效果，其步骤为：(1)确定领域范围；(2)确定领域中需要描述的概念，也称为类；(3)确定描述这些资源所需要的属性；(4)建立属性间的联系；(5)建立资源间的联系；(6)检查模式图的合理性，重复以上几步来修改模式图定义。

基于 Jena 的本体构建方法步骤如下：

(1)描述类。1)将领域中的资源归类，并抽象成概念，定义领域中的概念；2)按照领域中概念的相关性，将这些资源归类划分成不同集合；3)为每一个集合分配一个名称空间；4)为名称空间中的资源分配一个名字；5)建立这些概念之间的层次关系。

(2)描述属性。1)确定描述这些资源所需的属性；2)为每一个属性分配一个 URI，该 URI 也可以由名称空间和名字连接而成；3)描述该属性的特征(如对称的、可逆的)；4)属性可以拥有约束，包括属性值的类型约束、个数约束等；5)建立属性间的层次关系图。

(3)将属性和资源关联到一起。

(4)定义相关类的实例并填充其属性值，但不是必须的。

(5)加入维护此本体文档所需的元数据。

以上的步骤(2)～步骤(4)并没有严格的先后顺序，它们常常是相互嵌套、共同进行的。

2.2 方法实现算法

(1)描述类。本文示例代码创建了 3 个类：Animal 类，Dog 类和 Cat 类，并将 Dog 类和 Cat 类设置为 Animal 类的子类。Animal 类的 URI 标识由命名空间的字符串 exNs 与字符串"Dog"连接而成。

```
//在内存中建立本体模型，该模型使用 OWL 语言规则；
OntModel m = ModelFactory.createOntologyModel( OntModelSpec.OWL_MEM, null )；
//exNs 为名称空间的标识；
String exNs="http://cs.tongji.edu.cn/owl/ontologies/example/#"；
OntClass Animal = m.createClass( exNs + "Animal" )；
OntClass Dog = m.createClass( exNs + "Dog" )；
OntClass Cat= m.createClass( exNs + "Cat" )；
Animal.addSubClass(Dog)；
Animal.addSubClass(Cat)；
```

还可以将 Cat 类与 Dog 类设置为不相交类。为了方便用户的使用习惯，本文为 Dog 类建立了一个等价类 Doggie。

```
Dog.addDisjointWith(Cat)；
OntClass Doggie=m.createClass(exNs+"Doggie")；
Dog.addEquivalentClass(Doggie)；
```

(2)描述属性。在 OWL 本体中，属性分为对象属性和数据属性两种。示例代码创建了两个对象属性 hasDog 与 hasPet，并将 hasDog 设为 hasPet 的子属性。

```
OntClass People = m.createClass( exNs + "People" )；
OntClass Animal = m.createClass( exNs + "Animal" )；
ObjectProperty hasDog= m.createObjectProperty( exNs+ "hasDog" )；
ObjectProperty hasPet = m.createObjectProperty( exNs + "hasPet" )；
hasDog.addSuperProperty(hasPet )；
```

(3)将属性和类关联起来。本文将属性 hasPet 的定义域设为 People 类，值域设为 Animal 类。即 hasPet 只可以用于这样的 RDF 陈述[People 类实例，hasPet，Animal 类实例]。

```
hasPet.addDomain( People )；
hasPet.addRange( Animal)；
```

(4)创建实例。下面创建了一个叫 Charles 狗的 Dog 类实例。

```
Individual inst = m.createIndividual(exNs + " Charles", Dog )；
```

(5)加入维护本体的元数据，如版本。

```
Ontology ont=m.createOntology(exNs)；
ont.addProperty( RDFS.comment, "this is an example ontology about Pets Managing")；
ont.addProperty(RDFS.label,"Pet Management Ontology")；
```

本文所提出的本体构建方法在思路上较符合软件工程原理，较多地考虑了本体重用性，在代码实现上具有直观性、简明性和易于理解特点。下面以一个实例说明本方法在现实应用中的实用性和参考性价值。

3 实例

数字图书馆是新型网络资源组织模式，具有信息资源数字化、信息利用共享化、信息提供知识化等特点[4]。利用本体技术可以为数字图书资源提供智能搜索、管理功能。本文所提供的代码在Windows XP、J2SDK1.4.2 和Jena2.1 的环境下通过测试。

开始编写代码之前需做些准备工作，Jena可以从http://jena.sourceforge.net/downloads.html下载。下载文件包括一些例子，JavaDoc，源代码和jar 文件[3]。使用Jena之前需要设计类路径，方法是在操作系统的『控制面版』的『系统』的『环境变量』中创建一个CLASSPATH的变量，并将CLASSPATH设置为Jena2.1 安装目录下的lib目录中的一组jar文件名。jar文件名之间用“ ；”分开。

(1)描述图书馆领域类中的概念。可为书本、杂志、报纸等各类刊物建立一个共同的父类 Publication，该父类有所有刊物的共同属性。再依次建立 Publication 类的子类：Book 类，Magazine 类和 Paper 类等。使用的名称空间为 exNs。其领域结构如图 3 所示。

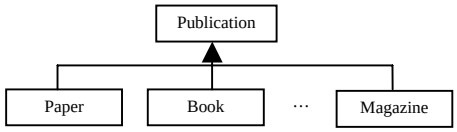

图 3 图书馆领域结构

```
String exNs="http://cs.tongji.edu.cn/owl/ontologies/example/#"；
OntModel m = ModelFactory.createOntologyModel( OntModelSpec.OWL_MEM, null )；
OntClass Publication=m.createClass(exNs+"Publication")；
```

(2)描述反映 Publication 特征的属性，本文采用 Dublin Core1.1 的元数据标准。Dublin Core 是国际组织 Dublin Core Metadata Initiative 拟定的用于标识电子资源的一种简要目录模式，有很好的实用性。

Jena 为 Dublin Core 专门提供了一个 DC 类，用户可直接

利用 DC.title 等属性。但 Jena 只是简单创建了 title、creator 等属性，而没有为这些属性制定任何约束，所以本文根据需要对这些属性的定义做了些修改。例如，可以为 title 属性建立一个基数约束，让每个 Publication 类的实例只可以有一个 title。

```
CardinalityRestriction titleR=m.createCardinalityRestriction(null, DC.title,1)；
Publication.addSuperClass(titleR)；
```

(3)将属性关联到类。本文为 relation 属性增加一个约束，让 relation 只可以应用于 Publication 类的实例，且其属性值也必须为 Publication 类的实例。

```
ObjectProperty relation = m.createObjectProperty( DC. relation. getNameSpace()+DC.relation.getLocalName())；
relation.addDomain(Publication)；
relation.addRange(Publication)；
```

此外，本文建立了一个 CategorySet(种类集合)的枚举类，所对应的实例为 Bookcategory，Magazinecategory，Papercategory 之一。现在可以为 Publication 添加一个 category(种类)属性，将其值设置为 CategorySet 实例。

```
Resource bookcategory = m.createResource( exNs + "book category" )；
Resource magazinecategory = m.createResource( exNs + "magazinecategory" )；
Resource papercategory = m.createResource( exNs + "paper category" )
RDFList cs = m.createList( new RDFNode[] {bookcategory, magazinecategory, papercategory} )；
EnumeratedClass CategorySet=m.createEnumerated Class(exNs+ "CategorySet",cs)；
categoryR=m.createAllValuesFromRestriction(null,category,Categ orySet)；
Publication.addSuperClass(m.createAllValuesFromRestriction(nul l,category,CategorySet))；
```

(4)利用 m.createIndividual(Resource cls)函数来创建 cls 类对应的实例。

(5)在此 OWL 本体文件加入关于维护此本体的元数据。

```
Ontology ont=m.createOntology(exNs)；
ont.addProperty( RDFS.comment, "this is an example ontology for the digital library")；
ont.addProperty(RDFS.label,"Publication Ontology")；
```

4 结束语

由于本体可以为网络资源提供确切的语义，因此一直被认为是语义网和信息自动化处理中的一项关键技术，但是本体构建还有许多需要研究和探索的地方。本文根据软件工程原理，提出了 OWL 本体构建的 5 个步骤，并给出了相应的基于 Jena 的实现算法。本文所提出的方法具有直观性、简明性和一定的实用性，实现了对本体的编程化构造和处理，可望对相关领域研究人员有所启迪和帮助。

参考文献

1 宋 炜, 张 铭. 语义网简明教程[M]. 上海: 高等教育出版社, 2004.

2 Jena 2——A Semantic Web Framework for Java[Z]. 2006-05-04. http://jena. sourceforge.net/index.html.

3 Verzulli J. Using the Jena API to Process RDF[Z]. 2001-05-23. http://www. xml.com/pub/a/2001/05/23/jena.html.

4 王丽华. 基于语义网的数字图书馆的关键技术[J]. 情报杂志, 2004, 25(4): 121-125.

(上接第 58 页)

下的信息就是掉电可保存的了。由于 yaffs 分区的容量很大，因此可以在此分区下保存大的文件。

2.4 混合文件系统集成

混合文件系统的集成就是将前面设计的 ramdisk 模块、jffs2 模块和 yaffs 模块集成到一个影像文件当中，使得系统只有一个文件系统影像文件，却可以使用 3 种文件系统。

集成的基本思路是将 ramdisk 作为系统的根文件系统，然后将 jffs2 模块和 yaffs 模块作为子文件系统挂载上，混合文件系统加载的过程如图 5 所示。

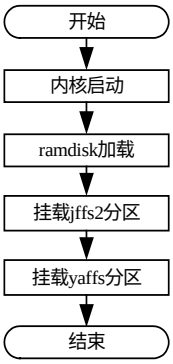

图 5 系统加载流程

自动挂载 jffs2 和 yaffs 分区需要修改根文件系统 ramdisk 下的文件 rc.local。添加

```
mount –t jffs2 /dev/mtdblock/3 /mnt/jffs2
mount –t yaffs /dev/mtdblock/4 /mnt/Nand1
mount –t yaffs /dev/mtdblock/5 /mnt/Nand2
mount –t yaffs /dev/mtdblock/6 /mnt/Nand3
mount –t yaffs /dev/mtdblock/7 /mnt/Nand4
```

当系统启动后，系统的目录包括/bin /sbin /usr/bin /usr/sbin 等位于 ramdisk 上，是不可以掉电保存的，这也起到了一定的保护作用，防止系统的崩溃。用户的编程数据，可以保存到/mnt/jffs2 分区中，做到实时的保存和修改。对于占用空间需要长期保存的数据文件，如数据库、图片、影音文件等，则可以保存到/mnt/Nand1 等 yaffs 分区。

3 结论

本文设计的混合文件系统充分应用了嵌入式设备中的 ram 和 flash 存储资源，做到了对数据的动态和静态存储。节约了系统的内存资源，和单一的 ramdisk 相比，更加合理地利用了系统的存储资源，特别是满足了对一些数据的永久性保存。本设计具有结构清晰、实现简单、性能优越的特点，目前已经作为某公司的核心产品之一，可以支持 1GB 以上的大容量 flash 存储。目前正在开发支持网络存储的新的文件系统。随着内存和 flash 存储的性价比的进一步提高以及网络技术的发展，在嵌入式设备上实现稳定可靠、功能强大的文件系统具有很好的研究价值和市场前景。

参考文献

1 Yaghmour K. 构建嵌入式 Linux 系统[M]. O'Reilly Taiwan, 译. 北京: 中国电力出版社, 2004-12.

2 Rubini A, Corbet J. Linux 设备驱动程序[M]. 魏永明, 骆 刚, 姜君, 译. 北京: 中国电力出版社, 2004-09.

3 H9200 上 jffs2 文件系统的移植[Z]. http://www.hyesco.com/form. Hyesco. H9200 用户手册[Z]. 北京恒颐高科技术有限公司, 2005.