让 Jena 帮我们想

----网页搜索　吴辉

谈到互联网的未来，人们自然的想到了语义网，虽然这个概念早在 1999 年就被伯纳斯－李先生提出，随后被学术界吵得沸沸扬扬，但是，真正基于语义网的应用似乎还处于孵化期。本文当然不想去解释什么是语义网、语义网的体系架构等等知识，本文将结合一个具体应用来说说语义网的一些特性，并由此希望能引起大家对语义网研究的兴趣和信心。

首先，我们来讲讲语义网的数据组织。目前比较常用的还是 RDF 规范，及其由此衍生的其他标准，例如 OWL 等。RDF(Resource Description Framework)是以 XML 为基础的，并结合 W3C 制定的一些规范，从而使其成为一个真正功能强大标记规范。

好，废话少说，下面我们就来建立一个星座的 RDF 领域知识图。

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xingzuo = "http://www.yahoo.com.cn/search/xingzuo/data/">
<xingzuo:Entity>
      <xingzuo:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">白羊座</xingzuo:name>
   <xingzuo:start rdf:datatype="http://www.w3.org/2001/XMLSchema#int">321</xingzuo:start>
   <xingzuo:end rdf:datatype="http://www.w3.org/2001/XMLSchema#int">420</xingzuo:end>
   <xingzuo:mstart rdf:datatype="http://www.w3.org/2001/XMLSchema#int">3</xingzuo:mstart>
   <xingzuo:mend rdf:datatype="http://www.w3.org/2001/XMLSchema#int">4</xingzuo:mend>
</xingzuo:Entity>
…
</rdf:RDF>
```

图 1　星座类知识图

熟悉 XML 的高手一看就知道，这其实就是一个 XML 的文档，只是有些标签采用了特定的规范。

例如数据类型，表示字符串“string”为

`rdf:datatype="http://www.w3.org/2001/XMLSchema#string"`

表示整数“int”为

`rdf:datatype="http://www.w3.org/2001/XMLSchema#int"`

在 RDF 的模型里，任何关系都是一个 Statement，每个 Statement 由三部分组成：subject、predicate 和 object。subject 和 object 通过 predicate 建立关系，其中 subject 是主语，object 是宾语。其他资料，大家可以参考中文版的 RDF 入门 http://wiki.w3china.org/cn/rdfprimer.htm，如果英语够厉害，不妨看看 http://www.w3.org/TR/rdf-primer/，此处不在累赘。

有了领域知识，我们就有了基础，呵呵，下面我们就开始来推理查询吧。目前，实现对 RDF 推理的免费 Reasoner 比较多，针对网上的评价和资源，我选择了 Jena，至于其他的引擎，大家可以 Yahoo(http://www.yahoo.cn)一下哦。唉，总算能进入正题了，现在就让我们来开始 Jena 之旅吧。

在使用之前，我们先简单介绍一下 Jena。Jena 是面向语义 Web 的应用开发包，包含的内容比较全面，推理机只是其中一部分，该推理机是针对本体的。至于什么是本体，大家也去 Yahoo 一下吧☺

Jena 推理的流程是：

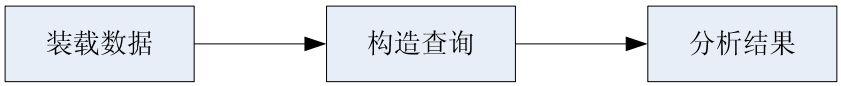

图 2 Jena 推理流程

下面对每个模块我们来分别介绍：

1. 装载数据

每个领域知识就是一个 Ontology Model，因此，我们需要将 RDF 表述的领域知识装载到一个 OntModel 中，代码如下：

```
private OntModel model;
…
FileInputStream file = new FileInputStream(filePath); //filePath 为领域库的保存路径
InputStreamReader in = new InputStreamReader(file, "UTF-8"); //文件为 UTF-8 编码
model = ModelFactory.createOntologyModel();
model.read(in, null);
```

图 3 装载 OntModel 代码片段

2. 构造查询

有了 OntModel，那么该领域的知识就在内存中构造成了一张图，现在我们要做的就是对这个图进行查询了。如果您属于数据库的 SQL 查询语言，那么对 OntModel 的查询对您就非常简单了，因为这儿用的是和 SQL 思想非常类似的 SPARQL 语言。

我们先来看看 SQL 和 SPARQL 的一段对比代码吧。

Select age From table1 Where name=”张三”	PREFIX test <http://test.cn/search/test/data/> Select ?age Where{?x test:name ?name . ?x test:age ?age. FILTER(?name = “张三”) }

图 4 SQL 与 SPARQL 语法对比

呵呵，是不是很相似啊！SPARQL 还有很多超妙的用法，别急着学那个，我还没有讲完呢☺（更多内容请参考：http://www.yahoo.cn/s?p=sparql%E4%BB%8B%E7%BB%8D&v=web&pid=hp）

好的，我们回头看看刚才那个星座类知识库中的 Statement 属性吧。从图 1 中我们可以看到，该图的属性有“name、start、end、mstart、mend”，其属性含义解释如下：

```
Name: 星座名称
Start: 该星座开始时间
End: 该星座结束时间
Mstart: 该星座开始月份
Mend: 该星座结束月份
```

图 5 星座类知识图属性解释

假设，我们现在的需求是“查找 9 月份的星座”。Ok，首先，我们分析一下这句话的含义：9 月份的日子可能出现的星座，由于星座的时间特殊性，那么可能是起始月在 9 月份，或者是终止月在 9 月份。那么这个 SPARQL 就可以写为：

```
PREFIX  xingzuo: http://www.yahoo.com.cn/search/xingzuo/data/
Select ?name
Where{
  ?x xingzuo:name ?name .
  ?x xingzuo:mstart ?mstart .
  ?x xingzuo:mend ?mend .
  FILTER(?mstart = 9 || ?mend = 9)
}
```

图 6 SPARQL 查询语句实例

上面只是 SPARQL 语法，那么，在程序中如何执行查询呢？下面的代码会告诉你：

```
String queryStr = …;
Query query = QueryFactory.create(queryStr);
QueryExecution qe = QueryExecutionFactory.create(query, model);
ResultSet results = qe.execSelect();
```

图 7 Jena 查询代码片段

这样的逻辑大家应该都不陌生，因为太想普通的数据库查询逻辑了☺

3. 分析结果

走完上面的两步，从图 7 中我们已经得到查询结果对象 results，我们想要的结果找到了，下面让我们来开始“收获”吧：

```
Vector mxz = new Vector();
while (results.hasNext())
{
    QuerySolution tqs = (QuerySolution) results.next();
    mxz.addElement(new String(tqs.getLiteral("name").getString()));
}
```

搞定。

好，现在我们已经通过一个实例介绍了 Jena 对 RDF 文件的查询和推理。如果你想玩更酷的功能，请参阅 Jena 的随包文档。