Digester 解析 XML 文档研究

Research on Parsing XML with Digester

徐秀华　汪诚波　（宁波 浙江大学宁波理工学院　315104）
毕硕本　（南京师范大学 地理科学学院 210097）

摘要：分析了目前解析 XML 文档基本方法的优缺点，并介绍了简单、高效解析 XML 文档的 Jakarta Commons Digester 方法。论述了 Digester 中常用的基本概念、实现步骤，最后用一个实例说明了 Digester 解析 XML 文档的实现过程。

关键词：XML　Digester　解析

1　概述

XML（extensible Markup Language，可扩展标记语言）作为一种简单、易读的数据表示形式，被看作企业间进行网络交换数据的最佳格式。它通过文档类型定义（DTD）或模式，使数据交换独立于各企业的数据库模式。

解析 XML 是 XML 应用中的关键一环。XML 文档解析通常有两种基本方法，即 DOM（Document Object Model，文档对象模型）和 SAX（Simple API for XML，XML 简单 API）。

1.1 DOM 方法

DOM 是维护和访问 XML 文档的应用程序接口（API），它定义了文档的逻辑结构以及存取和维护文档的方法。采用 DOM 方法进行解析时，XML 解析器读取整个 XML 文档，并在内存中生成一个 DOM Tree（树状表达方式），对 DOM Tree 的节点进行添加操作，实现对整个 XML 文档的全面、动态访问。

优点：概念清晰，结构良好，代码易读。

缺点：由于整个 XML 文档必须一次解析完成，占用较多的内存空间，解析时间长。

1.2 SAX 方法

采用 SAX 方法进行解析时，XML 解析器产生 SAX 事件，处理 SAX 事件。它的特点是不需要遍历整个文件而是用事件驱动来解析文件。

优点：解析 XML 文档的 SAX 方法，需要的代码量常常少于 DOM 的编码量，而且 SAX 速度快，占用内存空间少。

缺点：SAX 的概念不如 DOM 直观，代码的易读性不如 DOM，SAX 编程的结构性也通常不如 DOM。

DOM 和 SAX 方法各自都有优缺点，而 Apache Jakarta 的 Digester 结合了两者的优点，具有简单、高效、占用内存少等特点。Digester 最早出现在 Jakarta 的 Struts 中，后来随着 Struts 的发展以及其公用性而被提到 Commons 中独自立项。Digester 底层实现的是 SAX 解析，它通过为 SAX 事件提供高级的接口而大大简化了 SAX 方法的解析过程。该接口隐含了大量 XML 文件处理的复杂技术细节，使开发人员得以集中精力处理 XML 数据，而不是在如何解析文件本身的问题上花太多的时间。因此，Digester 经常被用于解析 XML 配置文件。

2　Digester 中的概念

为了更好地应用 Digester，需要深入理解三个重要概念：元素匹配模式，处理规则以及对象栈。

2.1 元素匹配模式

为了简化使用，Digester 通过元素匹配模式来定位要解析的 XML 元素标签。如果将 XML 文件结构看作为一颗树，那么每个标签的匹配模式就是从根元素到这个元素的路径。下面是一个 XML 层次结构的元素匹配模式的例子：

```
<? xml version = "1.0" ? >
<students >
    <student >
        <name >Tom </name >
        <course >Java </course >
    </student >
    <student >
        <name >Rose </name >
        <course >XML </course >
    </student >
</students >
```

每个标签与相应的匹配模式对应如表 1 所示。

2.2 处理规则

元素匹配模式用以识别什么时候采取行动，处理规则用以定义行动的内容。Digester 预定了很多处理规则，并允许用户进行自定义，自定义的处理规则是通过扩展 org.apache.commons.digester.Rule 类来实现。

从形式上讲，一个处理规则是一个 java 类，它扩展了 org. apache. commons. digester. Rule 类。每个处理规则实现下列的一个或几个事件处理方法（event method），当相应的模板匹配成功后，在已定义的某个时刻，这些事件方法会被触发。

表 1

标签	匹配模式
< students >	students
< student >	students/student
< name >	students/student/name
< course >	students/student/course

（1）begin（），在一个匹配元素被识别出后的“开始”时刻被调用，这个元素的所有属性放在一个数据结构中，并传递给 begin（）。

（2）body（），当元素的嵌套内容（如子元素）被识别出时被调用。在解析的过程中，前后的空白被去掉。

（3）end（），匹配元素的“结束”时刻被调用。如果子元素也需要匹配相关的规则，则这些规则的方法需都执行毕，才能达到该元素的“结束”时刻。

（4）finish（），解析结束时被调用，以提供给各个规则以清理临时数据的机会。

在设置 Digester 时，通过调用 addRule（）方法来注册一个特定的元素匹配模式以及相应的 Rule 类的实例。这个机制允许动态地生成 Rule 的实现。

此外，Digester 还提供一些处理常见情况的处理规则类，主要有：

① ObjectCreate，创建对象实例。

② SetProperties，将标签属性（Attribute）与要创建的对象的属性相关联。

③ SetNext，设置遇到下一个标签时的动作。

④ CallMethod，设置当匹配模式被找到时要调用的方法。

⑤ CallParam，设置对应的 callMethod 中指定方法所需要的参数值。

2.3 对象栈

对 Digester 技术最普通的应用，是用来动态创建一个由 Java 对象构成的树结构。各对象的属性以及对象间的关系，是基于 XML 文档的内容来设置的（XML 文档就是一棵树）。为实现这种应用，Digester 提供一个对象栈，在匹配模式识别后激活处理规则操作。对象栈的基本操作包括：

（1）clear（），清空栈的内容。

（2）peek（），返回对栈顶对象的引用。

（3）pop（），将栈顶对象弹出并返回。

（4）push（），将一个新的对象压入栈顶。

对象栈的运行过程如下：当识别出一个 XML 元素的“开始”时，将相关对象生成并压入栈顶，这个对象在处理该元素的子元素过程中一直在栈中，当所有子元素都处理完后，解析器遇到这个元素的“结束”时，则弹出此对象，并进行相关的处理。对象可以由处理规则或 Digester 中的相应方法压入和弹出对象栈。

3 应用实例

3.1 实现步骤

使用 Digester，通常按照以下步骤进行：

（1）建一个 org. apache. commons. digester. Digeter 实例。一个解析请求完成后，这个 Digester 可以被后面复用。

（2）根据需要设置一些配置属性（configuration properties），以控制下一步的解析操作。

（3）将一个或几个初始对象（initial object）压入 Digester 对象栈。

（4）注册所有的元素匹配模板（element matching pattern）。当一个模板从输入文档中被识别出后，与其相联系的处理规则（processing rules）被激活。

（5）调用 digester. parse（）方法，一个 XML 文档的引用（有多种方式供选择）要传给此方法，并需捕捉处理 IOException、SAXException 或处理过程中抛出的异常。

3.2 实现举例

下面是一个用于配置数据库的 XML 文档，该数据库配置文件定义了数据库提供商、数据库驱动、主机名、数据库名、用户名、密码等有关数据库配置方面的参数。

```
<? xml version = "1.0"  encoding = "UTF – 8" ? >
<jdbc >
    <vender > Oracle < /vender >
    <driver > oracle. jdbc. driver. OracleDriver < /driver >
    <url > jdbc: oracle: thin: @ xxh: 1521: xxh < /url >
    <host > hostname < /host >
    <dbname > dbname < /dbname >
    <username > username < /username >
    <password > password < /password >
< /jdbc >
```

在使用 Digester 之前，需要下载一些类包。主要有一个遵循 SAX2.0 或 Jaxp1.1 的 XML 解析器、Jakarta Commons 中的 beanutils、collections、logging 等类包。下面给出采用 Digester 实现解析 XML 文档的主体代码。

```
//导入 Digester 中需要的类声明
import org. apache. commons. digester. Digester;
```

```
import org. xml. sax. SAXException;
public class JdbcDigester
{
 public void run( ) throws IOException, SAXException
 {
  Digester digester = new Digester( );
  digester. push( this);
  //这个规则调用了 addJdbc 方法,该方法有 7 个参数
  digester. addCallMethod ( " jdbc" , " addJdbc" , 7);
  digester. addCallParam( " jdbc/vender" , 0);
  digester. addCallParam( " jdbc/driver" , 1);
  digester. addCallParam( " jdbc/url" , 2);
  digester. addCallParam( " jdbc/host" , 3);
  digester. addCallParam( " jdbc/dbname" , 4);
  digester. addCallParam( " jdbc/username" , 5);
  digester. addCallParam( " jdbc/password" , 6);
  //对数据库配置文件 jdbc. xml 解析
  digester. parse( " jdbc. xml" );
 }
 //addJdbc 方法
 public void addJdbc( String vender,
                      String driver,
                      String url,
                      String host,
                      String dbname,
                      String userName,
                      String password)
 {
  // 获得数据库参数,创建连接
  ...
 }
}
```

4 结束语

Jakarta Commons Digester 作为开放源码的 XML 文档解析 API,常被用于解析 XML 配置文件。本文中的实例采用 Digester 标准 API 进行解析,可以简单、高效地解析 XML 文档。Digester 还提供了 RuleSet 接口,允许用户通过扩展 RuleSetBase 类来开发规则集类,使得开发的规则集很容易在其他应用程序中直接复用;此外,可以使用日志类对 Digester 规则集进行调试,方便用户开发解析 XML 文档程序。

参考文献

1 Otis Gospodnetic, Parsing, indexing, and searching XML with Digester and Lucene, http://www-106.ibm.com/developerworks/j-ava/library/j-lucene, 2003.12.17.

2 Philipp K. Janert, Learning and Using Jakarta Digester, http://www.onjava.com/pub/a/onjava/2002/10/23/digester.html,2003.12.17.

3 Erik Swenson, Simplify XML file processing with the Jakarta Commons Digester, http://www.javaworld.com/javaworld/jw-10-2002/jw-1025-opensourceprofile.html, 2003.12.17.

4 Michael Morrison,et al. 著,陆新年、陆新宇等译,XML 揭密—入门、应用、精通[M],清华大学出版社,2001.6。