

附录C SAX 1.0 : XML简单API

本附录包含了SAX接口规范。它忠实地反映了 <http://www.megginson.com/sax/> 上的SAX规范，其中楷体字为编辑增加的注释。

在本附录中，类和接口是按照字母顺序依次描述的；在每个类中，方法也是按照字母顺序排列的。

SAX规范是公开免费发行的：上面列出的地址提供了版权声明。从本质上讲，它的版权政策如下：你可以随意进行处理和拷贝，但是我们对于由此引起的错误和遗漏概不负责。

SAX还包含三个“辅助类”：

- `AttributeListImpl`类是`AttributeList`接口的实现。
- `LocatorImpl`类是`Locator`接口的实现。
- `ParserFactory`类允许你在运行时加载以参数标识的解析器。

本附录没有包含这些辅助类的信息。可以通过访问 <http://www.megginson.com>，详细了解辅助类，并研究SAX样例应用程序。

C.1 类层次

```
class java.lang.Object
    interface org.xml.sax.AttributeList
    class org.xml.sax.helpers.AttributeListImpl
        (implements org.xml.sax.AttributeList)
    interface org.xml.sax.DTDHandler
    interface org.xml.sax.DocumentHandler
    interface org.xml.sax.EntityResolver
    interface org.xml.sax.ErrorHandler
    class org.xml.sax.HandlerBase
        (implements org.xml.sax.EntityResolver,
            org.xml.sax.DTDHandler,
            org.xml.sax.DocumentHandler,
            org.xml.sax.ErrorHandler)
    class org.xml.sax.InputSource
    interface org.xml.sax.Locator
    class org.xml.sax.helpers.LocatorImpl
        (implements org.xml.sax.Locator)
    interface org.xml.sax.Parser
    class org.xml.sax.helpers.ParserFactory
class java.lang.Throwable (implements java.io.Serializable)
class java.lang.Exception
class org.xml.sax.SAXException
class org.xml.sax.SAXParseException
```

C.1.1 `org.xml.sax.AttributeList`接口

`AttributeList`是出现在特定起始标记中的一组属性。在解析器提供的`DocumentHandler`接口

的startElement事件中，包含AttributeList信息。AttributeList基本上是元素的一组属性的名-值对；如果解析器分析了DTD，它还要提供每种属性类型的信息。

元素的属性接口规范

SAX解析器实现该接口，并将接口实例作为每个 startElement事件的第二个参数传递给 SAX 应用程序。

只有在startElement调用的范围内，解析器提供的实例才能返回有效的结果（为了保存结果以备将来使用，应用程序必须制作拷贝：AttributeListImpl辅助类提供了便利的构造器）。

AttributeList仅包含指定的或缺省的属性：它不包括 #IMPLIED属性。

SAX应用程序可以通过两种方式从 AttributeList中获取信息。首先，它可以对整个列表执行循环操作：

```
public void startElement (String name, AttributeList atts) {
    for (int i = 0; i < atts.getLength(); i++) {
        String name = atts.getName(i);
        String type = atts.getType(i);
        String value = atts.getValue(i);
        [...]
    }
}
```

(需要注意的是，如果列表中没有属性，getLength()将返回零。)

另外，应用程序还可以请求特定属性的值或类型：

```
public void startElement (String name, AttributeList atts) {
    String identifier = atts.getValue("id");
    String label = atts.getValue("label");
    [...]
}
```

AttributeListImpl辅助类提供了AttributeList的实现，解析器或应用程序编写者可以非常方便地使用它。

表 C-1

名 称	描 述
getLength	返回列表中包含的属性数
public int getLength()	SAX解析器可以以任意顺序提供属性，无论这些属性是按照什么顺序声明或指定的。属性数可以是零
	返回：
	列表中包含的属性数
getName	(根据位置)返回列表中指定属性的名称
public String getName(int index)	名称必须是唯一的：SAX解析器不允许多次包含相同的属性。列表 中将忽略没有值的属性（被声明为 #IMPLIED，且在起始标记中没有 指定值的属性）。
	如果属性名称有命名空间前缀，要保留该前缀。
	参数：
	index——属性在列表中的索引（从 0 开始）。
	返回：

(续)

名 称	描 述
<code>getType</code> <code>public String getType(int index)</code>	<p>被索引的属性的名称，如果索引超出有效范围，则返回空</p> <p>(根据位置) 返回列表中指定属性的类型</p> <p>属性类型可以是以下字符串：“CDATA”、“ID”、“IDREF”、“IDREFS”、“NMTOKEN”、“NMTOKENS”、“ENTITY”、“ENTITIES”或“NOTATION”(永远是大写)</p> <p>如果解析器没有读取属性的声明，或者解析器不报告属性的类型，则根据XML 1.0建议(3.3.3属性值的规格化)的规定返回值“CDATA”</p> <p>对于非表示法的枚举属性，解析器将返回类型值“NMTOKEN”</p> <p>参数：</p> <p>index——属性在列表中的索引(从0开始)</p> <p>返回：</p>
<code>getType</code> <code>public String getType(String name)</code>	<p>字符串形式的属性类型，如果索引超出有效范围，则返回空</p> <p>(根据名称) 返回列表中指定属性的类型</p> <p>与<code>getType(int)</code>的返回值相同</p> <p>如果文档中的属性名称有命名空间前缀，则应用程序在此也必须包含该前缀。</p> <p>参数：</p> <p>name——属性的名称</p> <p>返回：</p>
<code>getValue</code> <code>public String getValue(int index)</code>	<p>字符串形式的属性类型，如果索引超出有效范围，则返回空</p> <p>(根据位置) 返回列表中指定属性的值。</p> <p>如果属性值是名称记号列表(IDREFS、ENTITIES或NMTOKENS)，返回的字符串中将包含这些记号的序列，它们之间以空格分隔</p> <p>参数：</p> <p>index——属性在列表中的索引(从0开始)。</p> <p>返回：</p>
<code>getValue</code> <code>public String getValue(String name)</code>	<p>字符串形式的属性类型，如果索引超出有效范围，则返回空</p> <p>(根据名称) 返回列表中指定属性的值</p> <p>与<code>getValue(int)</code>的返回值相同</p> <p>如果文档中的属性名称有命名空间前缀，则应用程序在此也必须包含该前缀</p> <p>参数：</p> <p>name——属性的名称</p> <p>返回：</p>
	字符串形式的属性值，如果索引超出有效范围，则返回空

C.1.2 org.xml.sax.DocumentHandler接口

每个SAX应用程序都要包含实现该接口的类，它可以直接产生 `HandlerBase` 类的实例，或者继承 `HandlerBase` 类。第6章详细讨论了各种方法。

接收普通文档事件的通知

这是大多数 SAX 应用程序实现的主要接口：如果应用程序需要被告知基本解析事件，它实现该接口，并通过 setDocumentHandler 方法向 SAX 解析器注册实例。解析器使用该实例报告与文档相关的基本事件，例如：元素和字符数据的开始及结束。

在该接口中，事件的顺序至关重要，它反映了文档中信息的顺序。例如，元素的所有内容（字符数据、处理指令和 / 或子元素）将依次出现在 startElement 事件和相应的 endElement 事件之间。

不希望实现整个接口的应用程序编写者可以继承 HandlerBase 类，该类实现了缺省的功能；解析器编写者可以通过实例化 HandlerBase 类，获得缺省的处理程序。应用程序可以利用 setDocumentLocator 方法通过 Parser 提供的 Locator 接口获得任何文档事件的位置。

表 C-2

名 称	描 述
characters	接收字符数据的通知
public void characters(char ch[], int start, int length) throws SAXException	<p>Parser 将调用这个方法报告每个字符数据块。SAX 解析器可以在一个数据块中返回所有相邻的字符数据，或者将它们分为若干块；然而，任何一个事件中的所有字符必须来自同一外部实体，以便 locator 能够提供有价值的信息</p> <p>应用程序不能试图读取指定范围之外的数组，而且不能向数组写入数据</p> <p>某些解析器使用 ignorableWhitespace() 方法（而不是本方法）报告空白字符（验证有效性的解析器必须具备该功能）</p> <p>参数：</p> <p>ch——XML 文档中的字符</p> <p>start——数组中的起始位置</p> <p>length——从数组中读取的字符数</p> <p>抛出：SAXException</p> <p>任何 SAX 异常，可能包含其他异常</p>
endDocument	接收文档结束的通知。
public void endDocument() throws SAXException	<p>对于每个文档，SAX 解析器只能调用一次本方法，它将是解析过程中调用的最后一个方法。除非解析过程（由于不可恢复的错误）无法继续，或者达到输入的结尾，否则解析器不会调用本方法</p> <p>抛出：SAXException</p> <p>任何 SAX 异常，可能包含其他异常</p>
endElement	接收元素结束的通知。
public void endElement(String name) throws SAXException	<p>SAX 解析器将在 XML 文档中每个元素的结尾调用本方法；对于每个 endElement() 事件，都有相应的 startElement() 事件（即使元素为空）</p> <p>如果元素名称有命名空间前缀，则名称中仍然要保留该前缀</p> <p>参数：</p> <p>name——元素类型名称。</p> <p>抛出：SAXException</p>

(续)

名 称	描 述
<pre> ignorableWhitespace public void ignorableWhitespace(char ch[], int start, int length) throws SAXException </pre>	<p>任何SAX异常，可能包含其他异常</p> <p>接收元素内容中可忽略的空白通知</p> <p>验证有效性的解析器必须使用本方法报告每个可忽略的空白块（参见 W3C XML 1.0建议的 2.10节）：对于不验证有效性的解析器，如果它们能够解析并使用内容模型，也可以使用本方法</p> <p>SAX解析器可以在一个数据块中返回所有相邻的字符数据，或者将它们分为若干块；然而，任何一个事件中的所有字符必须来自同一外部实体，以便Locator能够提供有价值的信息</p> <p>应用程序不能试图读取指定范围之外的数组，而且不能向数组写入数据</p> <p>参数：</p> <p>ch——XML文档中的字符</p> <p>start——数组中的起始位置</p> <p>length——从数组中读取的字符数</p> <p>抛出：SAXException</p>
<pre> processingInstruction public void processingInstruction(String target, String data) throws SAXException </pre>	<p>任何SAX异常，可能包含其他异常</p> <p>接收处理指令的通知</p> <p>对于每个被发现的处理指令，Parser将调用此一次本方法：处理指令可以出现在主文档元素之前或之后</p> <p>SAX解析器不能使用本方法报告XML声明（XML 1.0的2.8节）或文本声明（XML 1.0的4.3.1节）</p> <p>参数：</p> <p>target——处理指令的目标</p> <p>data——处理指令数据，如果未提供任何处理指令，则返回空</p> <p>抛出：SAXException</p>
<pre> setDocumentLocator public void setDocumentLocator(Locator locator) </pre>	<p>任何SAX异常，可能包含其他异常</p> <p>接收用于定位SAX文档事件源的对象</p> <p>强烈建议（但并非绝对需要）SAX解析器提供Locator：</p> <p>如果它能够提供Locator，它必须在调用DocumentHandler接口中的任何其他方法之前，通过调用本方法为应用程序提供Locator</p> <p>Locator使得应用程序能够确定任何与文档相关的事件的结束位置，即使解析器没有报告错误。典型情况下，应用程序将使用该信息报告自己的错误（例如：与应用程序的商业规则不匹配的字符内容）。如果要将定位器作为搜索引擎，它返回的信息恐怕不够充足</p> <p>需要注意的是，只有在调用本接口的事件的过程中，定位器才能够返回正确的信息。应用程序不能在其他时刻使用该方法。</p> <p>参数：</p> <p>locator——能够返回任何SAX文档事件位置的对象</p>
<pre> startDocument public void startDocument() throws SAXException </pre>	<p>接收文档开始的通知</p> <p>对于每个文档，SAX解析器只能在调用本接口或DTDHandler中的其他方法（除了setDocumentLocator）之前调用一次本方法</p>

(续)

名 称	描 述
startElement public void startElement(String name, AttributeList atts) throws SAXException	<p>抛出：SAXException 任何SAX异常，可能包含其他异常</p> <p>接收元素开始的通知</p> <p>Parser将在XML文档中每个元素的开始处调用本方法； 对于每个startElement()事件，都有相应的endElement()事件（即使元素为空）。在相应的endElement()事件之前，元素的所有内容都要依次被报告</p> <p>如果元素名称有命名空间前缀，则名称中仍要保留该前缀。值得注意的是，本方法提供的属性列表只包含有显式值的属性（指定了属性值，或者有缺省的属性值）： #IMPLIED属性将被忽略</p> <p>参数： name——元素类型名称 atts——与元素相关的属性（如果存在的话）</p> <p>抛出：SAXException 任何SAX异常，可能包含其他异常</p>

C.1.3 org.xml.sax.DTDHandler接口

如果应用程序希望接收与DTD相关的事件，它应该实现本接口。SAX不提供DTD的完整细节，但是本接口具备该功能，因为否则就无法访问文档体中引用的表示法和未解析实体。表示法和未解析实体在XML中是非常特殊的，因此大多数SAX应用程序不需要使用本接口。

接收与DTD相关的基本事件通知

如果SAX应用程序需要有关表示法和未解析实体的信息，则它要实现该接口，并利用解析器的setDTDHandler方法向SAX解析器注册实例。解析器使用该实例向应用程序报告表示法和未解析实体声明。

SAX解析器可以以任何顺序报告这些事件，无论表示法和未解析实体是按照什么顺序声明的；然而，所有DTD事件必须在文档处理器的startDocument事件之后，第一个startElement事件之前报告。

信息存储的任务是由应用程序完成的（它可以以哈希表或对象树的形式保存信息）。如果应用程序遇到“NOTATION”、“ENTITY”或“ENTITIES”类型的属性，它能够利用本接口提供的信息寻找与属性值对应的实体和/或表示法。

HandlerBase类提供了本接口的缺省实现，它将事件简单地忽略。

表 C-3

notationDecl public void notationDecl(String name, String publicId, String systemId)	<p>接收表示法声明事件的通知</p> <p>如果在解析的后续过程中需要引用表示法，应用程序应该记录声明中相应的表示法</p> <p>如果提供了URL形式的系统标识符，SAX解析器在将它传递给应用程序之前必须完全解析它</p> <p>参数：</p>
---	--

(续)

throws SAXException	name——表示法的名称。 publicId——表示法的公共标识符，如果不存在公共标识符则为空 systemId——表示法的系统标识符，如果不存在系统标识符则为空 抛出：SAXException 任何SAX异常，可能包含其他异常
unparsedEntityDecl public void unparsedEntityDecl(String name, String publicId, String systemId, String notationName) throws SAXException	接收未解析实体声明事件的通知 本方法中的表示法名称对应于 notationDecl()事件报告的表示法。如果在解析的后续过程中需要引用该实体，应用程序应该记录声明中相应的实体 如果提供了URL形式的系统标识符，SAX解析器在将它传递给应用程序之前必须完全解析它 参数： name——未解析实体的名称。 publicId——实体的公共标识符，如果不存在公共标识符则为空 systemId——实体的系统标识符，如果不存在系统标识符则为空 notationName——相关表示法的名称 抛出：SAXException 任何SAX异常，可能包含其他异常

C.1.4 org.xml.sax.EntityResolver接口

当XML文档包含外部实体引用时，解析器通常会自动分析 URL：定位并解析相关文件。本接口允许应用程序超越该行为。举例来说，如果你希望从本地服务器获取实体的另一个版本，或者保存在数据库的实体目前缓存在内存中，或者实体引用了可变的信息，如：当前日期，对于以上情况，你都需要应用程序具备自主的控制能力。

当解析器需要获得实体时，它调用本接口，接口将返回一个InputSource对象。

用于解析实体的基本接口

如果SAX应用程序需要对外部实体进行定制的处理，它必须实现本接口，并通过解析器的setEntityResolver方法向SAX解析器注册实例。

此后，一旦遇到外部实体（包括外部 DTD子集和外部参数实体），在解析器包含它们之前，应用程序可以截获它们。

大多数SAX应用程序都没必要实现本接口，但是对于从数据库或其他特殊的输入源构建XML文档的应用程序，或者使用URI替代URL的应用程序来说，本接口是非常有用的。

以下解析程序将为应用程序提供系统标识符为“ http://www.myhost.com/today ”的实体的字符流：

```
import org.xml.sax.EntityResolver;
import org.xml.sax.InputSource;

public class MyResolver implements EntityResolver {
    public InputSource resolveEntity (String publicId, String systemId)
    {
        ...
    }
}
```



```
        if (systemId.equals("http://www.myhost.com/today")) {  
            // return a special input source  
            MyReader reader = new MyReader();  
            return new InputSource(reader);  
        } else {  
            // use the default behaviour  
            return null;  
        }  
    }  
}
```

应用程序也可以利用本接口将系统标识符重定向到本地 URI，或者在某类目录中寻找替代品（可能通过公共标识符获得所需的目录）。

HandlerBase类实现了本接口的缺省行为，它总是返回空（以请求解析器使用缺省的系统标识符）。

表 C-4

名 称	描 述
resolveEntity public InputSource resolveEntity(String publicId, String systemId) throws SAXException, IOException	<p>允许应用程序解析外部实体</p> <p>Parser在打开除了顶级文档实体之外的任何外部实体(包括：外部DTD子集，DTD中引用的外部实体，以及文档元素中引用的外部实体)之前，会调用本方法：应用程序可以请求解析器使用可选的URI或其他输入源解析实体</p> <p>应用程序编写者可以通过本方法将外部系统标识符重定向到可靠的和/或本地URI，在某类目录中寻找公共标识符，或者从数据库或其他输入源（例如：对话框）读取实体</p> <p>如果系统标识符是URL，SAX解析器在将它报告给应用程序之前必须完全解析它</p> <p>参数：</p> <p>publicId——所引用的外部实体的公共标识符，如果不存在公共标识符则为空</p> <p>systemId——所引用的外部实体的系统标识符</p> <p>返回：</p> <p>描述新的输入源的InputSource对象，如果要求解析器打开到系统标识符的常规URI连接，则返回空</p> <p>抛出：SAXException</p> <p>任何SAX异常，可能包含其他异常</p> <p>抛出：IOException</p> <p>Java IO异常，可能是由于InputSource创建新的InputStream或Reader引起的</p>

C.1.5 org.xml.sax.ErrorHandler接口

如果你需要对错误进行特别处理，可以在应用程序中实现本接口。HandlerBase类提供了缺省的实现。

SAX错误处理的基本接口

如果 SAX 应用程序需要实现定制的错误处理，它必须实现本接口，并通过解析器的 `setErrorHandler` 方法向 SAX 解析器注册实例。此后，解析器可以通过本接口报告所有的错误和警告。

解析器可以使用本接口取代抛出异常：应用程序将决定是否针对不同类型的错误和警告抛出异常。然而，在调用 `fatalError` 方法之后，解析器不必继续提供有价值的信息（换句话说，SAX 驱动类可以捕获异常并报告 `fatalError`）。

`HandlerBase` 类提供了本接口的缺省实现，它忽略了警告和可恢复的错误，对于致命错误，它抛出 `SAXParseException`。应用程序可以选择扩展该类，而不必完全从头开始实现 `ErrorHandler` 接口。

表 C-5

名 称	描 述
<code>error</code> <code>public void error (SAXParseException exception)</code> <code>throws SAXException</code>	接收可恢复的错误的通知。 它对应于 W3C XML 1.0 建议 1.2 节中“错误”的定义。 例如，验证有效性的解析器可以使用这个回调信号报告违反有效性约束的情况。缺省的行为是不采取任何操作 调用本方法之后，SAX 解析器必须继续提供常规的解析事件：应用程序应该仍然有可能继续处理文档直至文档的结尾。如果应用程序不能做到这一点，解析器应该报告致命错误，即使 XML 1.0 建议并不要求如此 参数： exception——封装在 SAX 解析异常中的错误信息 抛出：SAXException 任何 SAX 异常，可能包含其他异常
<code>fatalError</code> <code>public void fatalError(SAXParseException exception)</code> <code>throws SAXException</code>	接收不可恢复的错误的通知 它对应于 W3C XML 1.0 建议 1.2 节中“致命错误”的定义。例如，解析器可以使用这个回调信号报告违反格式正规约束的情况 应用程序必须假设解析器调用本方法之后文档是不重用的，它只是为了搜集更多的错误信息才应该继续处理文档：实际上，一旦调用本方法，SAX 解析器可以停止报告其他任何事件 参数： exception——封装在 SAX 解析异常中的错误信息 抛出：SAXException 任何 SAX 异常，可能包含其他异常
<code>warning</code> <code>public void warning(SAXParseException exception)</code> <code>throws SAXException</code>	接收警告的通知 SAX 解析器将使用本方法报告 XML 1.0 建议定义的既非错误，又非致命错误的情况。缺省的行为是不采取任何操作 在调用本方法之后，SAX 解析器必须继续提供常规的解析事件：应用程序应该仍然有可能继续处理文档直至文档的结尾 参数： exception——封装在 SAX 解析异常中的警告信息 抛出：SAXException 任何 SAX 异常，可能包含其他异常

C.1.6 org.xml.sax.HandlerBase类

HandlerBase类是由SAX本身提供的：它提供大多数方法的实现，否则这些方法都要由应用程序来实现。如果你在应用程序中创建HandlerBase类的子类，只需要编写具有非缺省行为的方法。

处理器的缺省基类

HandlerBase类实现了以下四个 SAX接口的缺省行为：EntityResolver、DTDHandler、DocumentHandler和ErrorHandler。

当应用程序编写者只需要实现部分接口时，可以扩展本类；当应用程序不提供处理器时，解析器编写者可以实例化本类，以提供缺省的处理器。

需要注意的是，是否使用本类是可选的。

在下面的描述中，只介绍了每个方法的行为。要了解它们的参数和返回值，参见相应的接口定义。

表 C-6

名 称	描 述
characters public void characters(char ch[], int start, int length) throws SAXException endDocument public void endDocument() throws SAXException	缺省情况下，不执行任何操作。应用程序的编写者可以重载本方法，以针对每个字符数据块执行特殊的操作（例如：将数据添加到节点或缓存中，或者将数据打印到文件中）
endElement public void endElement(String name) throws SAXException error public void error(SAXParseException e) throws SAXException fatalError public void fatalError(SAXParseException e) throws SAXException	接收文档结束的通知 缺省情况下，不执行任何操作。应用程序的编写者可以在一个子类中重载本方法，以便在文档的结束处执行特殊的操作（例如：结束树节点，或者关闭输出文件） 缺省情况下，不执行任何操作。应用程序的编写者可以在一个子类中重载本方法，以便在每个元素的结束处执行特殊的操作（例如：结束树节点，或者将输出信息写入文件） 缺省的实现不执行任何操作。应用程序的编写者可以在一个子类中重载本方法，以便针对每个错误执行特殊的操作，例如：在日志文件中插入错误信息，或者将它输出到控制台 缺省的实现抛出SAXParseException异常。如果应用程序的编写者需要针对每个致命错误执行特殊的操作（例如：将所有错误搜集到一个报告中），他可以在一个子类中重载本方法：无论如何，调用本方法时，应用程序必须停止所有常规的处理，因为此时文档不再可靠，解析器也不再报告解析事件

(续)

名 称	描 述
<code>ignorableWhitespace</code> <code>public void ignorableWhitespace(char ch[], int start, int length)</code> <code>throws SAXException</code>	缺省情况下, 不执行任何操作。应用程序的编写者可以重载本方法, 以便针对每个可忽略的空白块执行特殊的操作(例如: 将数据添加到节点或缓存中, 或者将数据打印到文件中)
<code>notationDecl</code> <code>public void notationDecl(String name, String publicId, String systemId)</code>	缺省情况下, 不执行任何操作。如果应用程序的编写者希望跟踪文档中声明的表示法, 他可以在一个子类中重载本方法
<code>processingInstruction</code> <code>public void processingInstruction(String target, String data)</code> <code>throws SAXException</code>	缺省情况下, 不执行任何操作。应用程序的编写者可以在一个子类中重载本方法, 以便针对每个处理指令执行特殊的操作, 例如: 设置状态变量, 或者调用其他方法
<code>resolveEntity</code> <code>public InputSource resolveEntity(String publicId, String systemId)</code> <code>throws SAXException</code>	总是返回空, 以便解析器使用XML文档中提供的系统标识符 本方法实现了SAX缺省的行为: 应用程序的编写者可以在一个子类中重载它, 以执行特殊的转换, 例如: 在某类目录中查找, 或者进行URI重定向。
<code>setDocumentLocator</code> <code>public void setDocumentLocator(Locator locator)</code>	缺省情况下, 不执行任何操作。如果应用程序的编写者希望保存定位器, 以便定位其他文档事件, 他可以在一个子类中重载本方法
<code>startDocument</code> <code>public void startDocument()</code> <code>throws SAXException</code>	缺省情况下, 不执行任何操作。应用程序的编写者可以在一个子类中重载本方法, 以便在文档的开始处执行特殊的操作(例如: 分配树的根节点, 或者创建输出文件)
<code>startElement</code> <code>public void startElement(String name, AttributeList attributes)</code> <code>throws SAXException</code>	缺省情况下, 不执行任何操作。应用程序的编写者可以在一个子类中重载本方法, 以便在每个元素的开始处执行特殊的操作(例如: 分配新的树节点, 或者将输出信息写入文件)
<code>unparsedEntityDecl</code> <code>public void unparsedEntityDecl(String name, String publicId, String systemId, String notationName)</code>	缺省情况下, 不执行任何操作。应用程序的编写者可以在一个子类中重载本方法, 以便跟踪文档中声明的未解析实体
<code>warning</code> <code>public void warning(SAXParseException e)</code> <code>throws SAXException</code>	缺省的实现不执行任何操作。应用程序的编写者可以在一个子类中重载本方法, 以便针对每个警告执行特殊的操作, 例如: 在日志文件中插入警告信息, 或者将它输出到控制台

C.1.7 org.xml.sax.InputSource类

InputSource对象代表XML文档或它所引用的外部实体的容器（从技术角度讲，主文档本身就是一个实体）。InputSource类是由SAX提供的：通常，应用程序将InputSource类实例化，并在其中保存输入信息的位置，解析器会询问它从何处获取输入信息。

InputSource对象通过以下三种方式为解析器提供输入：系统标识符（或URL）、Reader（它传递统一码字符流）或InputStream（它传递未解释的字节流）。

XML实体的单一输入源

SAX应用程序可以利用本类将有关输入源的信息封装在一个对象中，它可能包含公共标识符、系统标识符、字节流（或许采用特定的编码形式）和/或字符流。

应用程序可以通过以下两种方式将输入源传递给解析器：作为Parser.parse方法的参数，或者作为EntityResolver.resolveEntity方法的返回值。

SAX解析器将利用InputSource对象确定如何读取XML输入信息。如果有字符流，解析器将直接读取该字符流；否则，如果存在字节流，解析器将使用该字节流；如果两者均不存在，解析器将试图打开到由系统标识符标识的资源的URI连接。

InputSource对象属于应用程序：SAX解析器不能通过任何方式修改它（如果需要的话，解析器可以修改该对象的拷贝）。

如果你以Reader或InputStream的形式提供输入，最好同时提供系统标识符。这样，虽然解析器不会使用URI获取真正的XML输入，但是它可以用于诊断，更重要的，它有助于解析文档中的任何相对URI，例如：实体引用。

表 C-7

名 称	描 述
InputSource	零参数缺省构造器
public InputSource()	
InputSource	使用系统标识符创建新的输入源
public InputSource(String systemId)	应用程序也可以使用 setPublicId 包含公共标识符，或者通过 setEncoding 指定字符编码，如果字符编码已知的话 如果系统标识符是URL，它必须被完全解析。 参数： systemId——系统标识符（URI）
InputSource	使用字节流创建新的输入源
public InputSource(InputStream byteStream)	应用程序的编写者可以通过 setSystemId 提供用于解析相对URI的基地址，通过 setPublicId 包含公共标识符，并且/或者通过 setEncoding 指定对象的字符编码 参数： byteStream——包含文档的未经处理的字节流

(续)

名 称	描 述
InputSource public InputSource(Reader characterStream)	使用字符流创建新的输入源 应用程序的编写者可以使用 setSystemId() 提供用于解析相对 URI 的基地址，利用 setPublicId 包含公共标识符
setPublicId public void setPublicId(String publicId)	字节流中不能包含字节顺序标记 设置本输入源的公共标识符 公共标识符总是可选的：如果应用程序的编写者包含它，它将作为位置信息的一部分
getPublicId public String getPublicId()	参数： publicId——字符串形式的公共标识符 获取本输入源的公共标识符 返回： 公共标识符，如果没有提供公共标识符，则返回空
setSystemId public void setSystemId(String systemId)	设置本输入源的系统标识符 如果存在字节流或字符流，则系统标识符是可选的，但是它仍然有用，因为应用程序可以利用它解析相对 URI，或者在错误和警告消息中包含它（仅当没有指定字节流或字符流时，解析器才试图打开至 URI 的连接）
getSystemId public String getSystemId()	如果应用程序知道系统标识符指向的对象采用的字符编码，它可以使用 setEncoding 方法注册该编码 如果系统 ID 是 URL，它必须被完全解析 参数： systemId——字符串形式的系统标识符 获取本输入源的系统标识符 getEncoding 方法将返回系统标识符所指向的对象采用的字符编码，如果编码未知的话，它返回空 如果系统 ID 是 URL，它必须被完全解析 返回： 系统标识符
setByteStream public void setByteStream(InputStream byteStream)	设置本输入源的字节流 如果指定了字符流，SAX 解析器将忽略字节流，但是字节流的优先级高于打开 URI 连接 如果应用程序知道字节流的字符编码，它将使用 setEncoding 方法进行设置 参数： byteStream——包含 XML 文档或其他实体的字节流
getByteStream public InputStream getByteStream()	获取本输入源的字节流 getEncoding 方法将返回本字节流采用的字符编码，如果编码未知，它将返回空

(续)

名 称	描 述
setEncoding public void setEncoding(String encoding)	返回： 字节流，如果未提供字节流，则返回空 如果编码已知的话，设置字符编码 编码必须是XML编码声明可接受的字符串（参见XML 1.0建议的4.3.3节） 如果应用程序提供了字符流，本方法不会对应用程序产生任何影响 参数： encoding——描述字符编码的字符串
getEncoding public String getEncoding()	获取字节流或URI的字符编码 返回： 编码，如果未提供编码，则返回空
setCharacterStream public void setCharacterStream(Reader characterStream)	设置本输入源的字符流。 如果指定了字符流，SAX解析器将忽略字节流，也不会试图打开到系统标识符的URI连接 参数： characterStream——包含XML文档或其他实体的字符流
getCharacterStream public Reader getCharacterStream()	获取本输入源的字符流 返回： 字符流，如果未提供字符流，则返回空

C.1.8 org.xml.sax.Locator接口

应用程序可以利用本接口提供的方法确定源XML文档的当前位置。

用于将SAX事件与文档位置相关联的接口。

如果SAX解析器希望为SAX应用程序提供位置信息，它要实现本接口，并通过文档处理器的setDocumentLocator方法将实例传递给应用程序。应用程序可以使用该对象获得XML源文档中其他任何文档处理器事件的位置。

需要注意的是，只有在每个文档处理器方法的范围内，该对象返回的结果才有效：如果应用程序试图在其他情况下使用定位器，将会得到不可预期的结果。

定位器并不是SAX解析器必须提供的，但是强烈建议SAX解析器支持该接口。如果解析器提供定位器，必须在报告任何其他文档事件之前进行设置。如果应用程序接收到startDocument事件时尚未设置定位器，则应用程序假设没有定位器。

表 C-8

名 称	描 述
getPublicId public String getPublicId()	返回当前文档事件的公共标识符 返回： 包含公共标识符的字符串，如果没有公共标识符，则返回空

(续)

名 称	描 述
getSystemId public String getSystemId()	返回当前文档事件的系统标识符 如果系统标识符是URL，解析器必须在将它传递给应用程序之前，完全解析URL 返回： 包含系统标识符的字符串，如果没有系统标识符，则返回空
getLineNumber public int getLineNumber()	返回当前文档事件结束位置的行号。值得注意的是，它是与文档事件相关的文本之后的第一个字符所在的行的位置 返回： 行号，如果没有行号，返回 -1
getColumnNumber public int getColumnNumber()	返回当前文档事件结束位置的列号。值得注意的是，它是与文档事件相关的文本之后的第一个字符所在的列的位置。每行中第一列为位置 1 返回： 列号，如果没有列号，返回 -1。

C.1.9 org.xml.sax.Parser接口

所有SAX解析器都必须实现本接口。应用程序通过创建解析器（即：实现本接口的类）实例并调用parse()方法解析XML文档。

SAX解析器的基本接口

所有SAX解析器必须实现这个基本的接口：它允许应用程序为不同类型的事件注册处理器，并从URI或字符流初始化一次解析。

所有SAX解析器还必须实现零参数构造器（当然也允许有其他构造器）。

SAX解析器是可重用的，但不可重入：一旦一次解析成功完成，应用程序可以重用解析器对象（可能使用不同的输入源），但是在一次解析过程中，它不能循环调用 parse()方法。

表 C-9

名 称	描 述
parse public void parse(InputSource source) throws SAXException, IOException	解析XML文档。 应用程序可以使用本方法构造 SAX解析器，并从任何有效的输入源（字符流、字节流或 URI）开始解析XML文档 在解析过程中，应用程序不能引用本方法（对于更多的 XML文档，应用程序应该创建新的 Parser对象）。一旦解析完毕，应用程序可以重用该Parser对象，它可能使用不同的输入源 参数： source——XML文档顶级的输入源

(续)

名 称	描 述
	抛出：SAXException 任何SAX异常，可能包含其他异常。
	抛出：IOException 来自解析器的IO异常，可能是由于应用程序提供的字节流或字符流产生的
parse public void parse(String systemId) throws SAXException, IOException	依据系统标识符（URI）解析XML文档 通常，应用程序要依据系统标识符读取文档，本方法是这种情况的快捷方式。它等价于以下方法： parse(new InputSource(systemId)); 如果系统标识符是URL，应用程序在将它传递给解析器之前必须完全解析它 参数： systemId——系统标识符（URI）
setDocumentHandler public void setDocumentHandler(DocumentHandler handler)	抛出：SAXException 任何SAX异常，可能包含其他异常 抛出：IOException 来自解析器的IO异常，可能是由于应用程序提供的字节流或字符流产生的 允许应用程序注册文档事件处理器 如果应用程序不注册文档处理器，SAX解析器报告的所有文档事件都会被默默地忽略（这是HandlerBase实现的缺省行为） 在解析过程中，应用程序可以注册新的或其他处理器，SAX解析器必须立即用新的处理器进行解析 参数： handler——文档处理器
setDTDHandler public void setDTDHandler(DTDHandler handler)	允许应用程序注册DTD事件处理器 如果应用程序不注册DTD处理器，SAX报告的所有DTD事件都将被默默地忽略（这是HandlerBase实现的缺省行为） 在解析过程中，应用程序可以注册新的或其他处理器，SAX解析器必须立即用新的处理器进行解析 参数： handler——DTD处理器
setEntityResolver public void setEntityResolver(EntityResolver resolver)	允许应用程序注册定制的实体分析器 如果应用程序不注册实体分析器，SAX解析器将分析系统标识符，并打开到实体本身的连接（这是HandlerBase实现的缺省行为） 在解析过程中，应用程序可以注册新的或其他实体分析器，SAX解析器必须立即用新的分析器进行处理

(续)

名 称	描 述
setErrorHandler public void setErrorHandler(ErrorHandler handler)	参数： resolver——用于分析实体的对象 允许应用程序注册错误事件处理器 如果应用程序不注册错误事件处理器，SAX解析器报告的所有错误事件都将被默默地忽略，除了fatalError会抛出 SAXException（这是HandlerBase实现的缺省行为） 在解析过程中，应用程序可以注册新的或其他处理器，SAX解析器必须立即用新的处理器进行解析
setLocale public void setLocale(Locale locale)	参数： handler——错误处理器 允许应用程序为错误和警告请求适当的区域 SAX解析器不一定要为错误和警告提供本地化操作；然而，如果它们不支持所请求的区域，必须抛出SAX异常。在解析过程中，应用程序不能请求修改区域
throws SAXException	参数： locale——Java Locale对象。 抛出：SAXException 如果不支持所请求的区域，则抛出异常（使用以前的区域或缺省的区域）

C.1.10 org.xml.sax.SAXException类

SAXException类用于表示解析器或应用程序在处理过程中检测到的错误。

封装普通的SAX错误或警告

本类可以包含来自XML解析器或应用程序的基本错误或警告信息：解析器的编写者或应用程序的编写者能够通过创建该类的子类扩展它的功能。SAX处理器可以抛出该异常，或者任何作为它的子类的异常。

如果应用程序需要传递其他类型的异常，它必须将这些异常包含在SAXException或者从SAXException派生出的异常中。

如果解析器或应用程序需要包含有关XML文档中特定位置的信息，应该使用SAXParseException子类。

表 C-10

名 称	描 述
getMessage public String getMessage()	返回本异常的详细信息 如果有内嵌的异常，且SAXException本身没有详细的信息，本方法将从内嵌的异常中返回详细信息

(续)

名 称	描 述
getException public Exception getException()	返回： 错误或警告信息 如果存在内嵌的异常，则返回该异常 返回： 内嵌的异常，如果不存在内嵌的异常，则返回空
toString public String toString()	将本异常转换为字符串 返回： 本异常的字符串形式

C.1.11 org.xml.sax.SAXParseException类

扩展了SAXException类。

本异常类代表解析器或应用程序检测出的错误或警告情况。除了具备 SAXException的基本功能之外，SAXParseException能够获取源文档中出现错误的位置的信息。对于应用程序检测到的错误，可以通过Locator对象获得该信息。

封装XML解析错误或警告

本异常将包含用于定位源 XML 文档错误的信息。值得注意的是，虽然应用程序可以将 SAXParseException作为传递给 ErrorHandler接口中处理器的参数，但是实际上，应用程序不需要抛出异常；它可以简单地读取其中的信息，并采取其它操作。

由于本异常是SAXException的子类，因此它继承了SAXException包含其它异常的能力。

表 C-11

名 称	描 述
SAXParseException public SAXParseException(String message, Locator locator)	从消息和Locator中创建新的SAXParseException 当应用程序在 DocumentHandler回叫方法中创建自己的异常时，本构造器非常有用 参数： message——错误或警告消息 locator——错误或警告的 Locator对象
SAXParseException public SAXParseException(String message, Locator locator, Exception e)	在SAXParseException中包含当前的异常 当应用程序在 DocumentHandler回叫方法中创建自己的异常，且需要包含当前的不是 SAXException子类的异常时，本构造器非常有用 参数： message——错误或警告消息，如果使用内嵌异常中的消息，则为空 locator——错误或警告的 Locator对象 e——任何异常

(续)

名称	描述
<code>SAXParseException</code> <code>public SAXParseException(String message, String publicId, String systemId, int lineNumber, int columnNumber)</code>	创建新的SAXParseException 本构造器非常适用于解析器的编写者 如果系统标识符是URL，解析器在创建异常之前必须完全解析它 参数： message——错误或警告消息 publicId——产生错误或警告的实体的公共标识符 systemId——产生错误或警告的实体的系统标识符 lineNumber——导致错误或警告的文本的结尾的行号 columnNumber——导致错误或警告的文本的结尾的列号
<code>SAXParseException</code> <code>public SAXParseException(String message, String publicId, String systemId, int lineNumber, int columnNumber, Exception e)</code>	创建包含内嵌异常的新的SAXParseException 需要包含不是SAXException子类的异常时，解析器编写者可以使用本构造器 如果系统标识符是URL，解析器必须在创建异常之前完全解析它 参数： message——错误或警告消息，如果使用内嵌异常中的消息，则为空 publicId——产生错误或警告的实体的公共标识符 systemId——产生错误或警告的实体的系统标识符 lineNumber——导致错误或警告的文本的结尾的行号 columnNumber——导致错误或警告的文本的结尾的列号
<code>getPublicId</code> <code>public String getPublicId()</code>	e——嵌入本异常中的另一异常 获取产生异常的实体的公共标识符 返回： 包含公共标识符的字符串，如果不存在公共标识符，则返回空
<code>getSystemId</code> <code>public String getSystemId()</code>	获取产生异常的实体的系统标识符。需要注意的是，“实体”一词包含顶级XML文档 如果系统标识符是URL，它将被完全解析 返回： 包含系统标识符的字符串，如果不存在系统标识符，则返回空
<code>getLineNumber</code> <code>public int getLineNumber()</code>	产生异常的文本的结束位置的行号 返回： 代表行号的整数，如果没有行号，则返回-1
<code>getColumnNumber</code> <code>public int getColumnNumber()</code>	产生异常的文本的结束位置的列号。每行中的第一列为位置1 返回： 代表列号的整数，如果没有列号，则返回-1