# Introduction to JAXP

Java API for XML Processing (JAXP) 是使用以Java编程语言编写的应用程序处理XML数据。JAXP利用解析器标准Simple API for XML Parsing（SAX）和文档对象模型（DOM）以便你可以选择以事件流的或者构建表示对象的方式来解析数据。JAXP还支持可扩展样式表语言转换（XSLT）标准，使您能够控制数据的呈现，并使您能够将数据转换为其他XML文档或其他格式，例如HTML。JAXP还提供了命名空间支持，允许您使用可能具有命名冲突的DTD。最后，从版本1.4开始，JAXP实现了Streaming API for XML（StAX）标准。

旨在设计的最大灵活性，JAXP允许您在应用程序中使用任何符合XML的解析器。它通过所谓的可插拔层来实现，可以让您插入SAX或DOM API的实现。可插拔层还允许您插入XSL处理器，让您控制XML数据的显示方式。

## Overview of the Packages

SAX和DOM API分别由XML-DEV组织和W3C定义。定义这些API的库如下所示：

* javax.xml.parsers: JAXP API，为不同厂商的SAX和DOM解析器提供了一个通用接口。
* org.w3c.dom: 定义Document类（一个DOM）以及DOM的所有组件的类。
* org.xml.sax: 定义基本的SAX API。
* javax.xml.transform: 定义可将XML转换为其他表单的XSLT API。
* javax.xml.stream: 提供StAX指定转换API。

Simple API for XML(SAX)是一个事件驱动，序列访问的机制，它以一个元素接一个元素的方式进行处理。该级别的API将XML读取并写入数据存储库或Web。对于服务器端和高性能应用程序，您需要充分了解此级别。但是对于大多数应用来说，极少的理解就足够了。

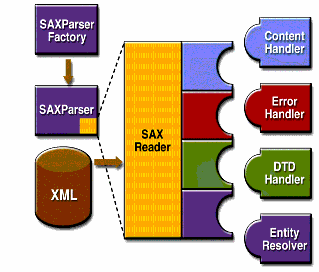
DOM API是一个更容易使用的API。它提供了一个熟悉的对象树结构。您可以使用DOM API来操纵其封装的应用程序对象的层次结构。DOM API是交互式应用程序的理想选择，因为整个对象模型都存在于内存中，这样用户可以访问和操作DOM对象。

另一方面，构建DOM需要读取整个XML结构并将对象树保存在内存中，因此它是CPU和内存密集型的。基于这个原因，SAX API倾向用于服务器端应用程序和数据过滤器，它们不需要内存中表示数据。

在javax.xml.transform中定义的XSLT API可以将XML数据写入文件或将其转换为其他形式。如本教程的XSLT部分所示，您甚至可以将其与SAX API结合使用，将旧数据转换为XML。

最后，javax.xml.stream中定义的StAX API提供了一个基于Java技术的流式，事件驱动，pull式解析的API，用来读取和写入XML文档。**StAX提供比SAX更简单的编程模型和比DOM更有效的内存管理。**

## Simple API for XML APIs

AX解析API的基本概要如图1-1所示。要启处理过程，SAXParserFactory类的实例用于生成解析器的实例。

解析器包装一个SAXReader对象。当解析器的parse()方法被调用时，reader调用应用程序中实现的几个回调方法中的一个。这些方法由接口ContentHandler，ErrorHandler，DTDHandler和EntityResolver定义。

以下是关键SAX API的摘要：

SAXParserFactory

SAXParserFactory对象创建由系统属性决定的解析器的实例。

SAXParser

SAXParser接口定义了几种parse()方法。 通常，需要传递给解析器一个XML数据源和一个DefaultHandler对象，用来处理XML并在handler对象中调用适当的方法。

SAXReader

SAXParser包装一个了SAXReader。通常情况下，您并不需要关心这一点，但是当你需要获取它的时候，使用SAXParser的getXMLReader()，这样您可以进行配置。实际上是SAXReader负责与你定义的SAX事件handler进行对话。

DefaultHandler

图中未显示，DefaultHandler实现了ContentHandler，ErrorHandler，DTDHandler和EntityResolver接口（里面是空方法），这样你可以只需要重写你需要的方法。

ContentHandler

当识别XML标签时，会调用诸如startDocument，endDocument，startElement和endElement之类的方法。该接口还定义了character()和processingInstruction()方法，当分析器遇到XML元素中的文本或内联处理指令时会调用这些方法。

ErrorHandler

当解析遇到各种错误时，会调用error()，fatalError(),和warning()方法。默认错误handler会为发生的致命错误抛出异常，并且忽略其他错误（包括验证错误）。这是您需要了解有关SAX解析器的一个原因，即使您正在使用DOM。有时，应用程序可能能够从验证错误中恢复。其他时候，它可能需要产生异常。为了确保正确的处理，您需要将自己的错误处理handler提供给解析器。

DTDHandler

定义的方法你通常不会被要求使用。用于处理DTD以识别和对未解析实体的声明采取行动。

EntityResolver

当解析器必须辨别由URI标识的数据时，会调用resolveEntity方法。在大多数情况下，URI只是一个URL，用于指定文档的位置，但在某些情况下，文档可能由网络空间中唯一的URN（公共标识符或名称）标识。除了URL之外，还可以指定公共标识符。然后，EntityResolver可以使用公共标识符而不是URL来查找文档，例如，如果文档存在，则访问文档的本地副本。

一个典型的应用程序至少实现了大部分ContentHandler方法。因为接口的默认实现DefaultHandler会忽略除致命错误之外的所有输入，所以一个健壮的实现也可能需要实现ErrorHandler方法。

### SAX Packages

SAX解析器在下表中列出的包中定义。

|  |  |
| --- | --- |
| **Packages** | **Description** |
| org.xml.sax | 定义了SAX接口。org.xml是由定义了SAX API的组织确定的包前缀名称。 |
| org.xml.sax.ext | 定义用于执行更复杂的SAX处理的SAX扩展，例如处理文档类型定义（DTD）或查看文件的详细语法。 |
| org.xml.sax.helpers | 包含一些使其更容易使用SAX帮助类。比如，通过定义了一个实现所有接口并且都是空方法的默认handler，以便您只需要重写那些实际要实现的方法。 |
| javax.xml.parsers | 定义SAXParserFactory类，返回SAXParser。还定义了用于报告错误的异常类。 |