Chapter 12: Simple API for XML (SAX)

Chapter 12 Objectives

- What is SAX?
- Where to download SAX and how to set it up
- How and when to use the primary SAX interfaces

What is SAX?

- Simple API for XML
  - Event-based, serial parser API for XML
  - Provide a standard way to parse XML documents
  - Analyze large documents more efficiently
- It is a *framework*
  - We build methods to be called → *callback* model
- Why do we need yet another parser?
  - DOM's problem

A Simple Analogy

- What does it mean by *event-based*?
- Like watching a train go by
  - Note the
    - Start of the train
      - Start of each car
        - Color
        - Length
        - Engineer
        - Baggage
        - Passengers
      - End of each car
    - End of the train

A Brief History of SAX

- Not a formal specification
- It is popular because it works!
- Since 1997
- Courtesy of David Megginson

Where to Get SAX

http://sourceforge.net/projects/sax

Xerces2-J
AElfred2
Crimson
Oracle
XP

Setting Up SAX

You will need 3 things.

SAX libraries
Parser
JDK

Receiving SAX Events

- Since it is a *interface*, we need

```
public class MyClass implements ContentHandler
```

- Problem?
  - Required to implement all the methods defined in the interface even not needed in application
- Solution – DefaultHandler

```
public class MyClass extends DefaultHandler
```

- DefaultHandler: *hard-working* class, also implement
  - ContentHandler
  - ErrorHandler
  - DTDHandler
  - EntityHandler

ContentHandler Interface

startDocument	processingInstruction
endDocument	startPrefixMapping
startElement	endPrefixMapping
endElement	setDocumentLocator
characters	skippedEntity
ignorableWhitespace	

Note that the startDocument event is NOT triggered by the root element. The start-tag of the root element does trigger an startElement event, but not the startDocument event.

Handling Element Events

```
public void startElement(String uri, String localName,
                         String qName, Attributes atts) throws SAXException
```

```
<myPrefix:myElement xmlns:myPrefix="http://example.com">
```

Parameter	**Value**
uri	http://example.com
localName	myElement
qName	myPrefix:myElement

Attributes

Method	Description
getLength	Determine the number of attributes available in the Attributes interface.
getIndex	Retrieve the index of a specific attribute in the list. The getIndex function allows you to look up the index by using the attribute's qualified name or by using both the local name and namespace URI.
getLocalName	Retrieve a specific attribute's local name by sending the index in the list.
getQName	Retrieve a specific attribute's qualified name by sending the index in the list.
getURI	Retrieve a specific attribute's namespace URI by sending the index in the list.
getType	Retrieve a specific attribute's type by sending the index in the list, by using the attribute's qualified name, or by using both the local name and namespace URI. If there is no Document Type Definition (DTD), this function will always return CDATA.
getValue	Retrieve a specific attribute's value by sending the index in the list, by using the attribute's qualified name, or by using both the local name and namespace URI.

Try It Out

Demo 1
Document and Element and Events Attributes Access

Handling Character Content

public void characters(char[] ch, int start, int len)
throws SAXException

- Text content stored in buffer
  ➔ Reuse and reduce memory use
- Buffer → string not as trivial
- Long text might trigger more than one character events
  ➔ Stored in more than one buffers

Handling Character Content

- **Ignorable Whitespace**

public void ignorableWhitespace(char[ ] ch, int start, int len)
throws SAXException

- **Skipped Entities**

public void skippedEntity(String name)
throws SAXException

Try It Out

Demo 2
Adding Colorful Characters

Handling Special Commands with Processing Instructions

public void processingInstruction(String target, String data)
throws SAXException

<?instructionForTrainPrograms blowWhistle?>
target: instructionForTrainPrograms
data: blowWhistle

<?xml-stylesheet type="text/xsl" href="courses.xsl"?>
target: xml-stylesheet
data: type="text/xsl" href="courses.xsl"

Namespace Prefixes

public void startPrefixMapping(String prefix, String uri)
throws SAXException

public void endPrefixMapping(String prefix) throws
SAXException

xmlns:example= http://example.com
prefix: example
uri: http://example.com

Providing the Location of the Error

Method	Description
getLineNumber	Retrieves the line number for the current event.
getColumnNumber	Retrieves the column number for the current event (the SAX specification assumes that the column number is based on right-to-left reading modes).
getSystemId	Retrieves the system identifier of the document for the current event. Because XML documents may be composed of multiple external entities, this may change throughout the parsing process.
getPublicId	Retrieves the public identifier of the document for the current event. Because XML documents may be composed of multiple external entities, this may change throughout the parsing process.

Try It Out

Demo 3
Which Stop is This?
Exception and setDocumentLocator

ErrorHandler Interface

Event	Description
warning	• Allows the parser to notify the application of a warning it has encountered in the parsing process. • Though the XML Recommendation provides many possible warning conditions, very few SAX parsers actually produce warnings.
error	• Allows the parser to notify the application that it has encountered an error. Even though the parser has encountered an error, parsing can continue. • ***Validation*** errors should be reported through this event.
fatalError	• Allows the parser to notify the application that it has encountered a fatal error and cannot continue parsing. • ***Well-formedness*** errors should be reported through this event.

Try It Out

Demo 4
To Err Is Human, To Handle Errors Is Divine

DTDHandler Interface

Event	Description
notationDecl	Allows the parser to notify the application that it has read a notation declaration.
unparsedEntityDecl	Allows the parser to notify the application that it has read an unparsed entity declaration.

DTD NOTATION declaration: Define the format of external non-XML file
Example: <!NOTATION gif PUBLIC “image/gif”>

EntityResolver Interface

Event	Description
resolveEntity	Allows the application to handle the resolution of entity lookups for the parser.

```
reader.setEntityResolver(this);
```

```
<!ENTITY train PUBLIC "-//TRAINS//freight cars xml 1.0//EN"
   "http://example.com/freighttrain.xml">
```


Features and Properties

Behaviors of SAX is controlled through setting features (such as validation) and properties (for registering extension interfaces)

```
public void setFeature(String name, boolean value)
  throws SAXNotRecognizedException, SAXNotSupportedException
```

Refer to feature list in the book.

Working with Properties

```
public void setProperty(String name, Object value)
  throws SAXNotRecognizedException, SAXNotSupportedException
public Object getProperty(String name)
  throws SAXNotRecognizedException, SAXNotSupportedException
```

Example:

```
reader.setProperty("http://xml.org/sax/properties/lexical-handler, lexHandler);
```

Property names in book.

Extension Interfaces: DeclHandler

Event	Description
attributeDecl	Allows the parser to notify the application that it has read an attribute declaration
elementDecl	Allows the parser to notify the application that it has read an element declaration.
externalEntityDecl	Allows the parser to notify the application that it has read an external entity declaration.
internalEntityDecl	Allows the parser to notify the application that it has read an internal entity declaration.

Extention Interfaces: LexicalHandler

Event	Description
comment	• Allows the parser to notify the document that it has read a comment. • The entire comment will be passed back to the application in one event call; it will not be buffered as it may be in the characters and ignorableWhitespace events.
startCDATA	• Allows the parser to notify the document that it has encountered a CDATA section start marker. • The character data within the CDATA section will always be passed to the application through the characters event.
endCDATA	• Allows the parser to notify the document that it has encountered a CDATA section end marker.
startDTD	• Allows the parser to notify the document that it has begun reading a DTD.
endDTD	• Allows the parser to notify the document that it has finished reading a DTD.
startEntity	• Allows the parser to notify the document that it has started reading or expanding an entity.
endEntity	• Allows the parser to notify the document that it has finished reading or expanding an entity.

Good SAX and Bad SAX

- **Good**
  - **Simple**
  - **Doesn't load whole document**
  - **Small footprint**
  - **Quick**
  - **Filtering**
- **Bad**
  - **No control over order**
  - **Intricate state keeping required**
  - **DOM is better at analyzing**

Consumers, Producers, and Filters

SAX can be used to

- Consume events (as in this module)
- Produce events (from non-XML)
- Filter events
  - Manipulate events as they pass through
  - XMLFilter interface in SAX
- Pipeline (from chain of filters)

Other Languages

Language	Available Interfaces
C++	• Xerces-C++, http://xml.apache.org/xerces-c • Microsoft Core XML Services (formerly MSXML), http://msdn.microsoft.com/xml • Arabica toolkit provides C++, http://www.jezuk.co.uk/cgi-bin/view/Arabica
Perl	http://perl-xml.sourceforge.net/libxml-perl/
Python	http://www.python.org/
Pascal	http://saxforpascal.sourceforge.net/
Visual Basic	MSXML, http://msdn.microsoft.com/library/en-us/xmlsdk/html/ac6be45a-177e-4b80-a918-dc73e357f7bb.asp
.NET	The System.Xml classes distributed with .NET http://saxdotnet.sourceforge.net
Curl	A web content management system with its own SAX bindings http://www.curl.com/