SAX
(Simple API for XML)

Sang Shin
JPassion.com
“Learn with Passion!”

Topics

- Parsing and Application
- SAX Event Model
- Error Handling
- JAXP 1.1
- When to Use SAX

Parsing and Application

Parsing and Application

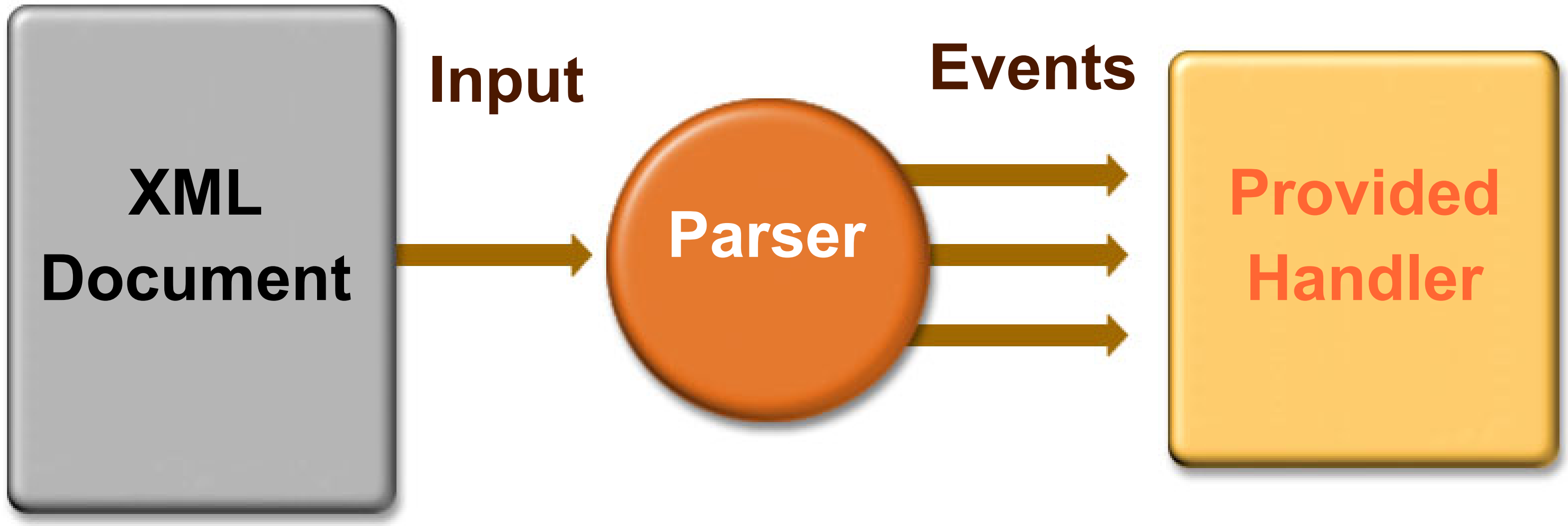
- Parsing includes
  - Well-formed'ness checking & Validating (against XML schema)
  - Reading elements and their values
- Application uses parsing for
  - Manipulating (XML document)
  - Creating (XML document)
  - Writing and Sending (XML document)

SAX Event Model

SAX Features

- Event-driven
  - You provide event handlers
- Fast and lightweight
  - Document does not have to be entirely in memory
- Sequential read access only
- One-time access
- Does not support modification of document

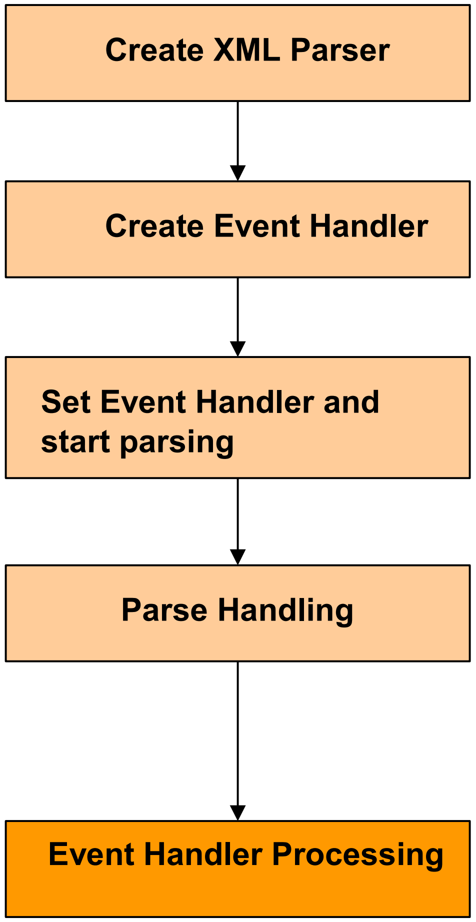
SAX Operational Model



SAX Programming

- Collection of Java interfaces and classes
  - org.xml.sax package
- Interfaces
  - Parser
    - *XMLReader*
  - Event handlers
    - *ContentHandler*

SAX Programming Procedures



```
XMLReader parser =
XMLReaderFactory.createXMLReader();
```

```
myHandler handler = new myHandler();
```

```
parser.setContentHandler(handler)

parser.parse(args[0]);
```

SAX parser calls methods on the event handler

```
public void startDocument(){
    System.out.println(“XML Document Start”);
}
```

XMLReader Interface

- Represents SAX parser
  - SAX2 parser implementation has to implement this interface
- Methods for
  - Registering event handlers
  - Initiating parsing
  - Configuring features and properties
    - Example: Validation on and off

XMLReader Interface

```
public interface XMLReader{
   public boolean getFeature (String name)
      throws SAXNotRecognizedException, SAXNotSupportedException;
   public void setFeature (String name, boolean value)
      throws SAXNotRecognizedException, SAXNotSupportedException;
   public Object getProperty (String name)
      throws SAXNotRecognizedException, SAXNotSupportedException;
   public void setProperty (String name, Object value)
      throws SAXNotRecognizedException, SAXNotSupportedException;
   public void setEntityResolver (EntityResolver resolver);
   public EntityResolver getEntityResolver ();
   public void setDTDHandler (DTDHandler handler);
   public DTDHandler getDTDHandler ();
   public void setContentHandler (ContentHandler handler);
   public ContentHandler getContentHandler ();
   public void setErrorHandler (ErrorHandler handler);
   public ErrorHandler getErrorHandler ();
   public void parse (InputSource input) throws IOException, SAXException;
   public void parse (String systemId) throws IOException, SAXException;
}
```

XMLReader Instance

- Concrete implementation instance “bound” to *XMLReader* interface
- Has to be created before parsing
- Gets created by using static method of *createXMLReader()* method of factory class *XMLReaderFactory*

XMLReader Example

```
XMLReader parser = null;
try {
    // Get SAX parser instance reading org.xml.sax.driver
    // system property.
    parser = XMLReaderFactory.createXMLReader();

    // Parse the document

} catch(SAXException ex){
    // Couldn’t create XMLReader
    // either because org.xml.sax.driver system property
    // was not set or set incorrectly.
}
```

Setting Features

- *setFeature(String, boolean)* method of *XMLReader* interface
- Features
  - General features
  - SAX features
  - DOM features

General Features (They are not resolvable URL's)

- http://apache.org/xml/features/validation/schema
  - When true, turn on XML schema support
- http://apache.org/xml/features/continue-after-fatal-error

SAX Features

- http://xml.org/sax/features/validation
  - When true, validate the document
- http://xml.org/sax/features/namespaces
  - When true, this feature indicates that the startElement( ) and endElement( ) methods provide namespace URIs and local names for elements and attributes.
- http://xml.org/sax/features/namespace-prefixes
  - When true, this feature indicates that xmlns and xmlns:prefix attributes will be included in the attributes list passed to startElement( ).

Example

```
XMLReader parser = null;
try {
    // Create an instance of Apache's Crimson SAX parser
    parser = XMLReaderFactory.createXMLReader();

    // Set features
    parser.setFeature("http://xml.org/sax/features/validation",
                                          true);

    // Parse the document

}catch(SAXException ex){
}
```

Parse Methods

- *void parse(String uri) throws SAXException, IOException*
- *void parse(InputSource source) throws SAXException, IOException*

Example

```
XMLReader parser = null;
try {
    parser = XMLReaderFactory.createXMLReader();

    // Parse the document
    parser.parse("http://www.slashdot.org/slashdot.xml");

    // Capture SAX events

}catch(SAXException ex){
    // exception occurs maybe because document
    // is malformed
}
```

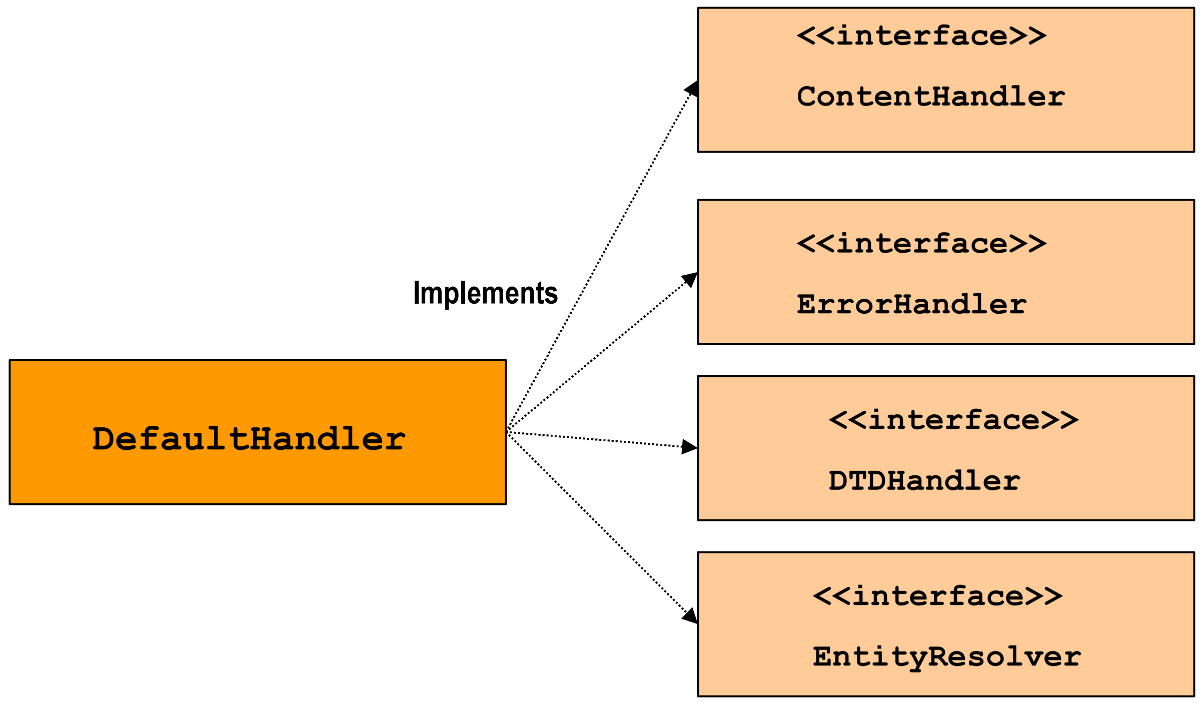
Example

```
XMLReader parser = null;
try {
    parser = XMLReaderFactory.createXMLReader();

    // Parse the document in File URI form
    parser.parse("file:/tmp/people.xml");

    // Capture SAX events

}catch(SAXException ex){
    // exception occurs maybe because document
    // is malformed
}
```

SAX Event Handlers

- Interfaces
  - *ContentHandler*
  - *ErrorHandler*
  - *DTDHandler*
  - *EntityResolver*
  - *Attributes*
- Class
  - *DefaultHandler* (Utility class)

SAX Event Handlers



ContentHandler Interface

```
public interface ContentHandler{
void startDocument () throws SAXException;
void endDocument() throws SAXException;
void startElement(String namespace, String name, String qName,
Attributes atts) throws SAXException;
void endElement(String namespace, String name, String qName) throws
SAXException;
void characters(char [ ] ch, int start, int length) throws SAXException;
void ignorableWhiteSpace(char [ ]ch, int start, int length) throws
SAXException;
void processingInstruction(String target, String data) throws
SAXException;
void setDocumentLocator(Locator locator);
void startPrefixMapping(String prefix, String uri) throws SAXException;
void endPrefixMapping(String prefix) throws SAXException;
void skippedEntity(String name) throws SAXException;
}
```

Simple SAX Example: Parser

```
XMLReader parser = null;
try {
    // Create XML (non-validating) parser
    parser = XMLReaderFactory.createXMLReader();
    // Create event handler
    myContentHandler handler = new myContentHandler();
    parser.setContentHandler(handler);
    // Call parsing method
    parser.parse(args[0]);
}catch(SAXException ex){
    System.err.println(ex.getMessage());
}catch(Exception ex){
    System.err.println(ex.getMessage());
}
```

Simple SAX Example: Event Handler

```
class myContentHandler implements ContentHandler {

 // ContentHandler methods
 public void startDocument(){
    System.out.println(“XML Document START”);
 }
 public void endDocument(){
    System.out.println(“XML Document END”);
 }
 public void startElement(String namespace, String name, String qName,
                               Attributes atts){
    System.out.println(“<“ + qName + “>”);
 }
 public void endElement(String namespace, String name, String qName){
    System.out.println(“</“ + qName + “>);
 }
 public void characters(char[] chars, int start, int length){
    System.out.println(new String(chars, start, length);
 }
```

Character Data

- Character data
  - void characters(char [ ] ch, int start, int length) throws SAXException
- Parsers are allowed to break up character data any way desired
- Character data are in Unicode regardless of encoding scheme specified in XML document

Attributes Interface

```
public interface Attributes{
public abstract int getLength();
public abstract int getIndex(String qName);
public abstract int getIndex(String namespace, String name)
public abstract String getLocalName(int index)
public abstract String getQName(int index)
public abstract String getType(int index)
public abstract String getType(String qName)
public abstract String getType(String namespace, String name)
public abstract String getValue(String qName)
public abstract String getValue(String namespace, String name)
public abstract String getValue(int index)
public abstract String getURI(int index)
}
```

Locator Interface

- Tells application where events occurred

```
public interface Locator{
public int getLineNumber();
public int getColumnNumber();
public String getPublicId();
public String getSystemId();
}
```

Locator Interface

- SAX parser passes implementation instance of Locator interface to the **ContentHandler.setDocumentLocator()**
  - It should be saved to a local reference if the application needs it

Locator Example

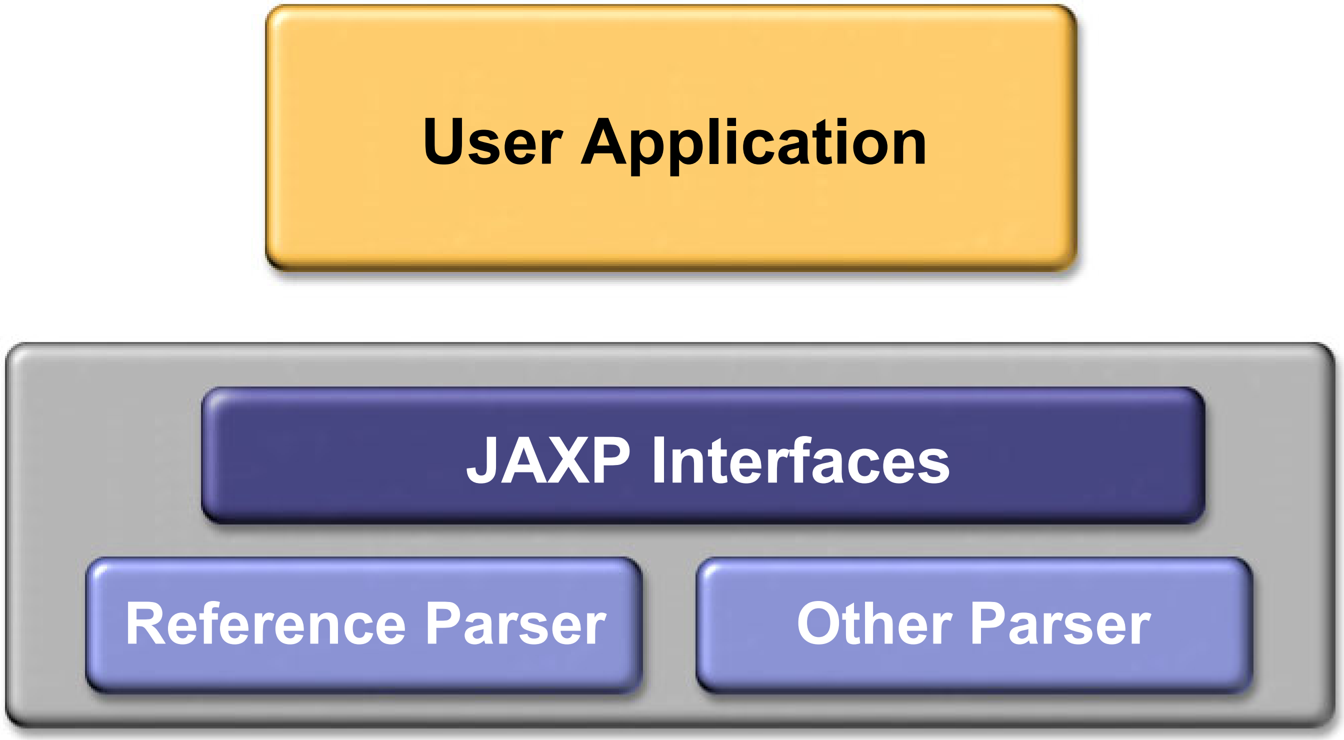
```
Locator loc;
public void setDocumentLocator(Locator loc){
  this.loc = loc;
}
public void startElement(String namespace, String name, String qName,
Attributes a){
  System.out.println(name);
  System.out.println(“ line: “ +  loc.getLineNumber());
  System.out.println(“ ID: “ + loc.getSystemId());
}
```

JAXP 1.1

JAXP 1.1

- A thin and lightweight Java API for parsing and transforming XML documents
- Allows for pluggable parsers and transformers
- Allows parsing of XML document using:
  - Event-driven (SAX 2.0)
  - Tree based (DOM Level 2)

JAXP: Pluggable Framework for Parsers and Transformers



When to Use SAX

Benefits Of SAX

- It is very simple
- It is very fast
- Useful when custom data structures are needed to model the XML document
- Can parse files of any size without impacting memory usage
- Can be used to gather a subset of a document’s information

Drawbacks Of SAX

- SAX provides read-only access
- No random access to documents
- Searching of documents is not easy

Learn with Passion!
JPassion.com