XML Programming: SAX

Andy Clark

17 Apr 2002

SAX Design Premise

- Generic method of creating XML parser, parsing documents, and receiving document information
- “Streaming” information set
  - Application registers handlers
  - Parser “pushes” information to handlers
    - Serial (“as you see it”) notification
  - Application only uses information it needs

org.xml.sax.XMLReader (1 of 2)

- Settings
  - void setFeature(String featureId, boolean state);
  - boolean getFeature(String featureId);
  - void setProperty(String propertyId, Object value);
  - Object getProperty(String propertyId);
- Parsing
  - void parse(String systemId);
  - void parse(InputSource inputSource);

org.xml.sax.XMLReader (2 of 2)

- Handlers
  - void setContentHandler(ContentHandler handler);
  - ContentHandler getContentHandler();
  - void setDTDHandler(DTDHandler handler);
  - DTDHandler getDTDHandler();
  - void setErrorHandler(ErrorHandler handler);
  - ErrorHandler getErrorHandler();
- Entity resolver
  - void setEntityResolver(EntityResolver resolver);
  - EntityResolver getEntityResolver();

Creating a SAX Parser

- org.xml.sax.helpers.XMLReaderFactory
  - XMLReader createXMLReader();
    - Creates default SAX parser
    - Search method
      - **System property: “org.xml.sax.driver”**
      - **Jar Services: “META-INF/services/org.xml.sax.driver”**
      - **Distribution specific fallback**
  - XMLReader createXMLReader(String className);
    - Creates SAX parser by name

Parsing a Document

- Instantiate parser
  - XMLReader reader = XMLReaderFactory.createXMLReader();
- Configure settings
  - reader.setFeature("http://xml.org/sax/features/namespaces", true);
  - reader.setFeature("http://xml.org/sax/features/validation", false);
- Parse document
  - reader.parse("document.xml");

org.xml.sax.ContentHandler (1 of 2)

- void setDocumentLocator(Locator locator);
- void startDocument();
- void endDocument();
- void processingInstruction(String target,
  String data);
- void skippedEntity(String name);

org.xml.sax.ContentHandler (2 of 2)

- void startPrefixMapping(String prefix, String uri);
- void endPrefixMapping(String prefix);
- void startElement(String uri, String localName, String qname, Attributes attrs);
- void endElement(String uri, String localName, String qname);
- void characters(char[ ] ch, int offset, int length);
- void ignorableWhitespace(char[ ] ch, int offset, int length);

Receiving SAX Events (1 of 4)

- Implementing an element counter

```
01  public class ElementCounter extends DefaultHandler {
02      private int elements;
03      public void startDocument() {
04          elements = 0;
05      }
06      public void startElement(String uri, String localName,
07                               String qname, Attributes attributes) {
08          elements++;
09      }
10      public void endDocument() {
11          System.out.println("elements: "+elements);
12      }
13  }
```

Receiving SAX Events (2 of 4)

- Implementing an element counter

```
01  public class ElementCounter extends DefaultHandler {
02      private int elements;
03      public void startDocument() {
04          elements = 0;
05      }
06      public void startElement(String uri, String localName,
07                               String qname, Attributes attributes) {
08          elements++;
09      }
10      public void endDocument() {
11          System.out.println("elements: "+elements);
12      }
13  }
```

Receiving SAX Events (3 of 4)

- Implementing an element counter

```
01  public class ElementCounter extends DefaultHandler {
02      private int elements;
03      public void startDocument() {
04          elements = 0;
05      }
06      public void startElement(String uri, String localName,
07                               String qname, Attributes attributes) {
08          elements++;
09      }
10      public void endDocument() {
11          System.out.println("elements: "+elements);
12      }
13  }
```

Receiving SAX Events (4 of 4)

- Registering custom content handler

```
01    XMLReader reader = XMLReaderFactory.createXMLReader();
02    reader.setFeature("http://xml.org/sax/features/namespaces", true);
03    reader.setFeature("http://xml.org/sax/features/validation", false);
04    reader.setContentHandler(new ElementCounter());
05    reader.parse("document.xml");
```

- Input
  - <name> <given>Andy</given> <family>Clark</family> </name>
- Output
  - elements: 3

Receiving Errors (1 of 4)

- Implementing error handler

```
01      public class ErrorPrinter implements ErrorHandler {
02         public void warning(SAXParseException e) {
03            System.out.println("[warning] "+e.getMessage());
04         }
05         public void error(SAXParseException e) {
06            System.out.println("[error] "+e.getMessage());
07         }
08         public void fatalError(SAXParseException e) throws SAXException {
09            System.out.println("[fatal error] "+e.getMessage());
10       throw e;
11    }
12 }
```

Receiving Errors (2 of 4)

- Implementing error handler

```
01      public class ErrorPrinter implements ErrorHandler {
02         public void warning(SAXParseException e) {
03            System.out.println("[warning] "+e.getMessage());
04         }
05         public void error(SAXParseException e) {
06            System.out.println("[error] "+e.getMessage());
07         }
08         public void fatalError(SAXParseException e) throws SAXException {
09            System.out.println("[fatal error] "+e.getMessage());
10       throw e;
11    }
12 }
```

Receiving Errors (3 of 4)

- Implementing error handler

```
01  public class ErrorPrinter implements ErrorHandler {
02      public void warning(SAXParseException e) {
03          System.out.println("[warning] "+e.getMessage());
04      }
05      public void error(SAXParseException e) {
06          System.out.println("[error] "+e.getMessage());
07      }
08      public void fatalError(SAXParseException e) throws SAXException {
09          System.out.println("[fatal error] "+e.getMessage());
10          throw e;
11      }
12  }
```

Receiving Errors (4 of 4)

- Registering custom error handler

```
01    XMLReader reader = XMLReaderFactory.createXMLReader();
02    reader.setFeature("http://xml.org/sax/features/namespaces", true);
03    reader.setFeature("http://xml.org/sax/features/validation", true);
04    reader.setErrorHandler(new ErrorPrinter());
05    reader.parse("document.xml");
```

- Input
  - <name> <given>Andy</given> <family>Clark</family> </name>
- Output
  - [error] Document root element "name", must match DOCTYPE root "null".
  - [error] Document is invalid: no grammar found.

Resolving Entities (1 of 4)

- Implementing entity resolver

```
01  public class ResolverPrinter implements EntityResolver {
02      public InputSource resolveEntity(String publicId, String systemId) {
03          System.out.println("publicId: "+publicId);
04          System.out.println("systemId: "+systemId);
05          return null;
06      }
07  }
```

Resolving Entities (2 of 4)

- Implementing entity resolver

```
01      public class ResolverPrinter implements EntityResolver {
02         public InputSource resolveEntity(String publicId, String systemId) {
03       System.out.println("publicId: "+publicId);
04       System.out.println("systemId: "+systemId);
05       return null;
06    }
07  }
```

Resolving Entities (3 of 4)

- Implementing entity resolver

```
01      public class ResolverPrinter implements EntityResolver {
02         public InputSource resolveEntity(String publicId, String systemId) {
03       System.out.println("publicId: "+publicId);
04       System.out.println("systemId: "+systemId);
05       return null;
06    }
07  }
```

  - *Note:* Always set the system identifier on the input source returned from your custom entity resolver.

Resolving Entities (4 of 4)

- Registering custom entity resolver

```
01    XMLReader reader = XMLReaderFactory.createXMLReader();
02    reader.setFeature("http://xml.org/sax/features/namespaces", true);
03    reader.setFeature("http://xml.org/sax/features/validation", false);
04    reader.setEntityResolver(new ResolverPrinter());
05    reader.parse("document.xml");
```

- Input
  - <!DOCTYPE root SYSTEM 'grammar.dtd'> <root/>
- Output
  - publicId: null
  - systemId: file:///D:/xml/examples/grammar.dtd

Useful Links

- SAX
  - http://sax.sourceforge.net/

XML Programming: SAX

Andy Clark