Java API para el procesado de documentos XML

(JAXP)

Bibliografía

- Todos los tutoriales de J2SE disponibles en:

  https://docs.oracle.com/javase/tutorial/

- Tutorial JAXP:

  https://docs.oracle.com/javase/tutorial/jaxp/index.html

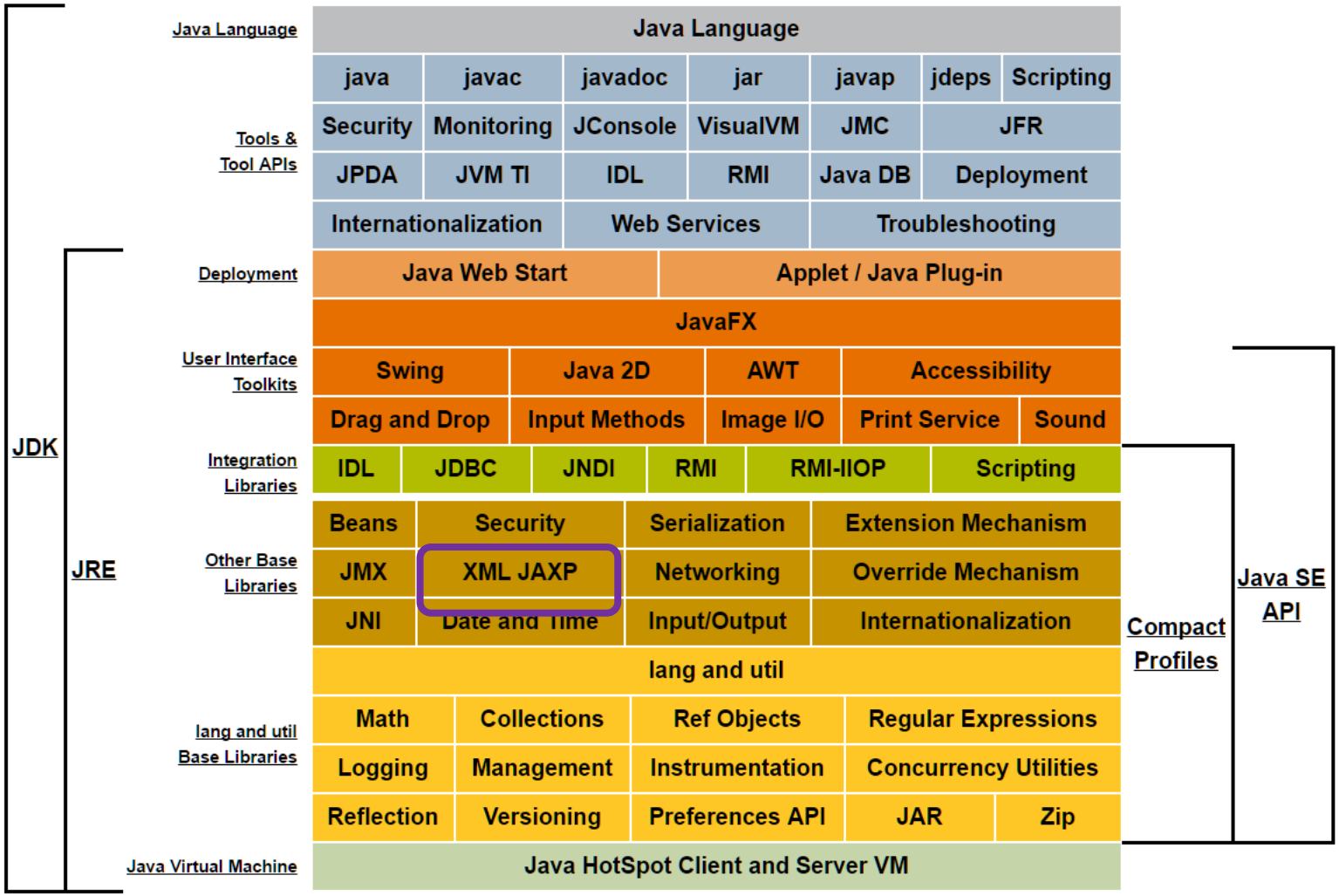
- Beginning XML, 5th Edition. Disponible en O'Reilly. Capítulo 11
- Sax Proyect

JAXP. Introducción

"The Java API for XML Processing (JAXP) is for processing XML data using applications written in the Java programming language. JAXP leverages the parser standards Simple API for XML Parsing (SAX) and Document Object Model (DOM) so that you can choose to parse your data as a stream of events or to build an object representation of it. JAXP also supports the Extensible Stylesheet Language Transformations (XSLT) standard, giving you control over the presentation of the data and enabling you to convert the data to other XML documents or to other formats, such as HTML. JAXP also provides namespace support, allowing you to work with DTDs that might otherwise have naming conflicts. Finally, as of version 1.4, JAXP implements the Streaming API for XML (StAX) standard."

Introduction to JAXP

JAXP en el diagrama conceptual J2SE



http://www.oracle.com/technetwork/java/javase/tech/index.html

JAXP. Introducción

- Permite manipular documentos XML mediante programas escritos en Java.
- Las bibliotecas que define son:
  - `java.xml.parser`: interfaz común (para todos los fabricantes) para intérpretes (*parsers*) SAX y DOM.
  - `org.w3c.dom`: define la clase documento (DOM) y todas las clases de los componentes de un DOM.
  - `org.xml.sax`: API SAX.
  - `javax.xml.transform`: API XSLT para transformar XML.
  - `javax.xml.stream`: API que proporciona transformaciones específicas StAX.

Paquetes includios en JAXP

- javax.xml.datatype
- javax.xml.namespace
- javax.xml.parsers
- javax.xml.stream
- javax.xml.stream.events
- javax.xml.stream.util
- javax.xml.transform
- javax.xml.transform.dom
- javax.xml.transform.sax
- javax.xml.transform.stax
- javax.xml.transform.stream
- javax.xml.validation
- javax.xml.xpath
- org.w3c.dom
- org.w3c.dom.ranges
- org.w3c.dom.traversal
- org.w3c.dom.bootstrap
- org.w3c.dom.events
- org.w3c.dom.ls
- org.xml.sax
- org.xml.sax.ext
- org.xml.sax.helpers

JAXP. APIs

- Simple API for XML (SAX): permite procesar un documento XML elemento a elemento, de manera **orientada a eventos**, mediante la serialización del contenido del documento XML.
- Document Object Model (DOM): permite manipular una estructura XML como si fuera un árbol de objetos almacenados en memoria.
- XSLT API: permite realizar transformaciones.
- StAX API: permite parsear y modificar XML streams como eventos usando un cursor.

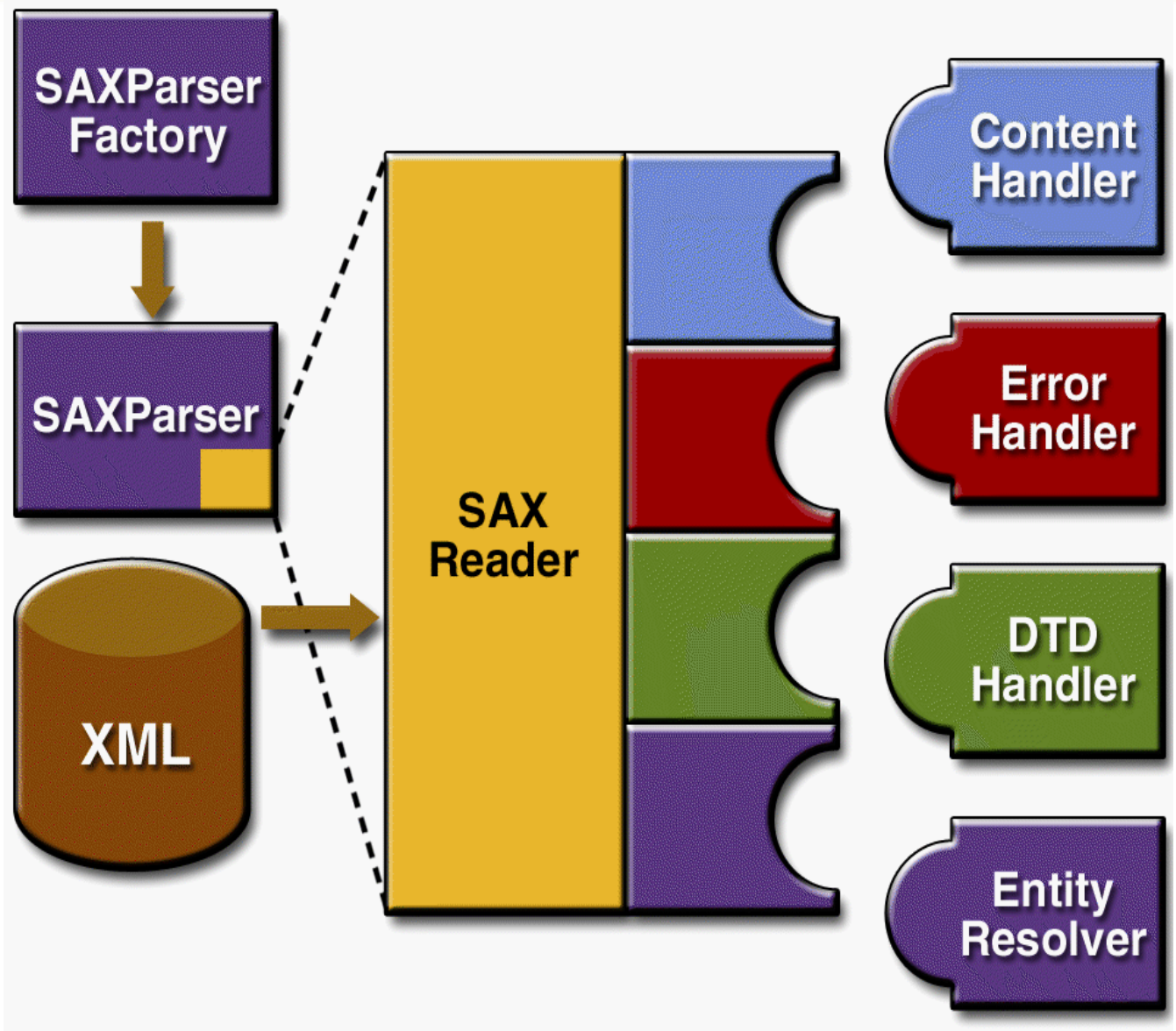
SAX

(Simple API for XML)

¿Cómo funciona SAX?

- Serializa el contenido de un documento XML y los envía a la aplicación.
- Crea un parser que sirve para procesar un documento XML.
- Permite modificar el formato.
- No construye una representación en memoria del contenido del documento XML.
- Es rápido en su funcionamiento y consume pocos recursos de memoria.
- Utiliza un modelo “push” para el manejo de eventos, por tanto el parser es el que tiene que mantener el estado después de cada tratamiento de un evento.

SAX. Arquitectura.



SAX. Introducción

- Al principio, una instancia de la clase `SAXParserFactory` es usada para generar una instancia del parser.

- El parser envuelve un objeto SAXParser. Cuando el metodo *parse()* del parser es invocado, se invoca uno de los métodos definidos por los interfaces `ContentHandler`, `ErrorHandler`, `DTDHandler` y `EntityResolver`.

SAX. Puntos clave.

- `SAXParserFactory:` crea una instancia del parser.
- `SAXParser:` define varias clases de métodos parse(). Se pasa un XML y el objeto DefaultHandler al parser, el cual procesa el XML invocando a los métodos del objeto manejador (handler).
- `SAXReader:` es el que está en comunicación con los manejadores de eventos definidos por el usuario.
- `DefaultHandler;` implementa los cuatro interfaces para que sólo haga falta redefinir los métodos necesarios.

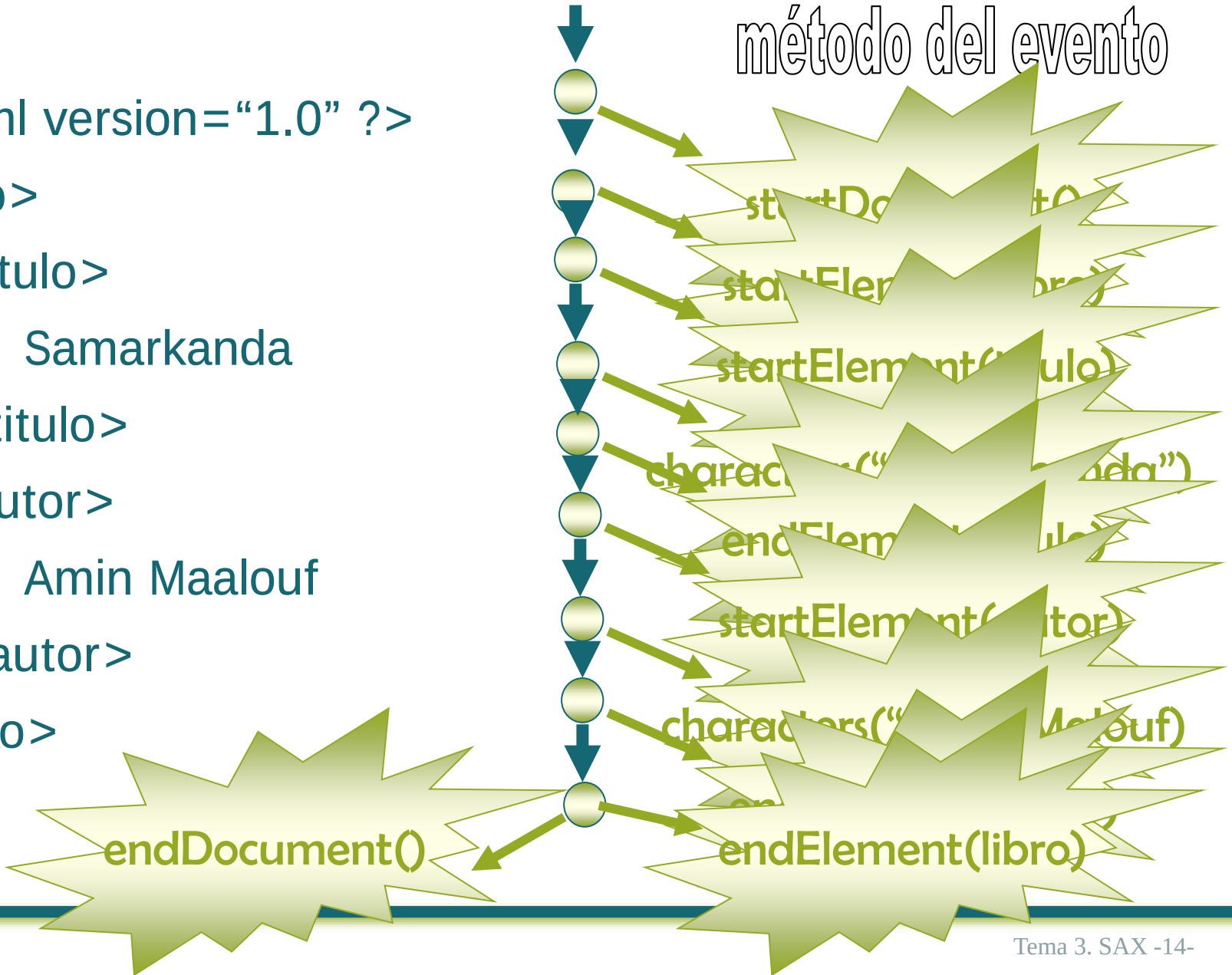
SAX. Puntos clave.

- `ContentHandler:` contiene los métodos invocados cuando sucede un evento.
- `ErrorHandler:` contiene los métodos invocados cuando el parser provoca un error.
- `DTDHandler:` usado en el procesamiento de DTD.
- `EntityResolver:` usado cuando el parser debe identificar datos mediante su URL.

Una aplicación típica se hace redefiniendo los métodos de `ContentHandler`.

SAX. Eventos. Interfaz contentHandler.

```
<? xml version="1.0" ?>
<libro>
    <titulo>
        Samarkanda
    </titulo>
    <autor>
        Amin Maalouf
    </autor>
</libro>
```



SAX. Paquetes.

`org.xml.sax`	Define los interfaces SAX
`org.xml.sax.ext`	Para procesado SAX avanzado (p..e. analizar un DTD)
`org.xml.sax.helpers`	Contiene las clases que hacen más fácil usar SAX
`java.xml.parsers`	Define la clase SAXParserFactory, que devuelve el SAXParser.

SAX. Paquetes. Java Docs

`org.xml.sax`	https://docs.oracle.com/javase/7/docs/api/org/xml/sax/package-summary.html
`org.xml.sax.ext`	https://docs.oracle.com/javase/7/docs/api/org/xml/sax/ext/package-summary.html
`org.xml.sax.helpers`	https://docs.oracle.com/javase/7/docs/api/org/xml/sax/helpers/package-summary.html
`java.xml.parsers`	https://docs.oracle.com/javase/7/docs/api/javax/xml/parsers/package-summary.html

Ejemplo SAX

```
import java.io.File;
import java.io.IOException;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;


public class Echo {
   //código en las siguiente diapositivas
}
```

Ejemplo SAX. Creación e invocación del saxParser

```
public static void main(String[] args) {
   if (args.length != 1) {
                System.err.println("Usage:cmd filename");
                System.exit(1);
   }
   try {
     SAXParserFactory factory = SAXParserFactory.newInstance();
     factory.setNamespaceAware(true);
     SAXParser saxParser = factory.newSAXParser();
     ManejadorXML manejadorXML = new ManejadorXML();
     saxParser.parse( new File(args[0]), manejadorXML);
   } catch (SAXException | ParserConfigurationException |
                IOException e) { e.printStackTrace(); }
}
```

Ejemplo SAX. Redefinición métodos DefaulHandler()

```
static class ManejadorXML extends DefaultHandler {
    private StringBuilder sb = new StringBuilder();
    @Override
    public void startDocument() throws SAXException {
            super.startDocument();
            System.out.println("Comienza el documento");
    }

    @Override
    public void endDocument() throws SAXException {
             super.endDocument();
             System.out.println("Finaliza el documento");
    }
```

Ejemplo SAX. Redefinición métodos DefaulHandler()

```
@Override
public void startElement(String namespaceURI,
                         String localName, String qName,
                         Attributes atts) throws SAXException {
   super.startElement(namespaceURI, localName, qName, atts);
   System.out.println( "SAX Event: START ELEMENT[ " +
                       localName + " ]" );
   for ( int i = 0; i < atts.getLength(); i++ ){
      System.out.println( " ATTRIBUTE: " + atts.getLocalName(i)
                          +  " VALUE: " + atts.getValue(i) ); }}
@Override
public void endElement(String uri, String localName,
                       String qName) throws SAXException {
        super.endElement(uri, localName, qName);
        System.out.println( "SAX Event: END ELEMENT[ " +
                            localName + " ]" );}
```

Ejemplo SAX. Redefinición métodos DefaulHandler()

```
    @Override
     public void characters(char[] ch, int start, int length)
      throws SAXException {
           super.characters(ch, start, length);
           System.out.print( "SAX Event: CHARACTERS[ ");
          sb.append(ch,start,lenght);
          System.out.print(sb.toString());
           System.out.println( " ]");
          sb.setLength(0);
     }
}
```

SAX

(Simple API for XML)