Introduction to JAXP

Integrating XML with Java

History

- V1.0: KhanhLT.
- V2.0: DoanNX (20100227).

Contents

- Overview of XML.
- Parsers and Parsing.
- JAXP Interfaces.

Time: 45’

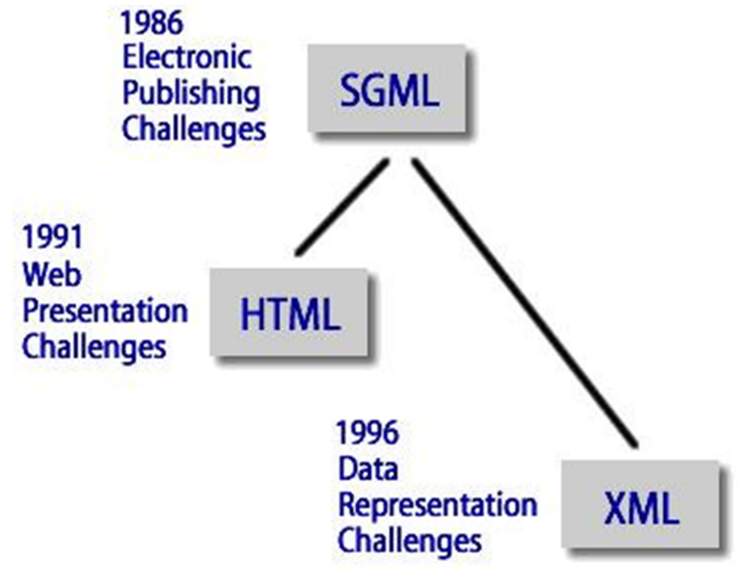
Contents

Overview of XML

- The need creating and presenting data in an interchangeable format progress from *Standard Generalized Markup Language* (SGML) to *Hypertext Markup Language* (HTML) then *Extension Makeup Language* (XML).
- SGML
  - Is a metalanguage can generate/define another language.
  - Possessed the capability of storing huge amount of data.
  - The process of handling heavy data was very complex.
  - Proved unsuitable for interchange of data between application.

Overview of XML

- In HTML
  - Drew many concepts from SGML.
  - Proved to be advantageous over SGML.
  - Was not really useful in creating efficient and smart searches on voluminous data.



Overview of XML

- XML
  - Text-based markup language that enables to store data in a structured format by using meaningful tags.
  - Purpose of XML is to store, transfer and describe the data.
  - XML is a cross platform, hardware, software and device independent markup language.
  - XML file is well-formed file.
    - Has an unique root node (DocumentElement).
    - Each opening tag must correspond with the same closing tag.
    - XML tags must be case-sensitive.
    - The child element must be placed completely in its parent element.
    - The value of attribute in XML must be placed in quotes/apostrophes.

Overview of XML

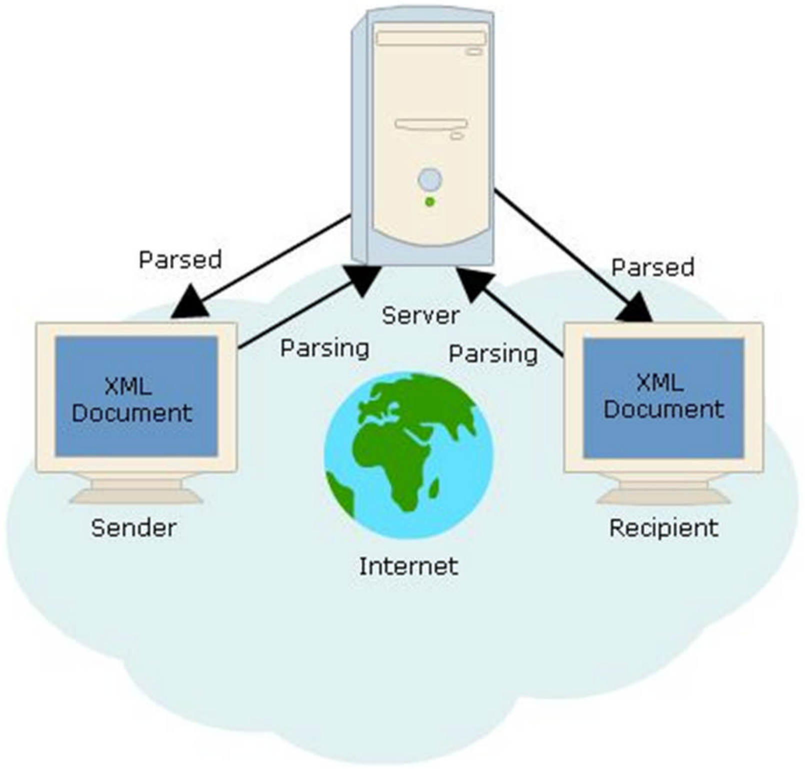
- XML (cont)
  - Advantages
    - Domain Specific Vocabulary – user defined tags.
    - Data Interchange between different computer system.
    - Enables Smart Searches – Titanic.
    - User-selected view of data – CSS, XSL.
    - Granular updates.

Overview of XML

- Roles of XML in Java
  - XML is the natural choice for developing enterprise level web application using Java.
  - Developers can implement the platform independent feature of the Java programming language to develop applications and exchange application data using XML.
  - The advantages of developing web applications using XML are:
    - Supports exchange of data between heterogeneous databases and systems.
    - Distributes data processing load to the web browser.
    - Integrates Java servers with web browsers.

Contents

- Overview of XML.
- Parsers and Parsing.
- JAXP Interfaces.



Parsers and Parsing

- XML Parsing
  - Process data using programs (parsers).
  - These programs are able to extract and manipulate data in XML documents.
  - Benefits:
    - Language Independent: XML is written with Chinese, French, Italian or English.
    - Code Independent: focuses on the syntax & the data present in that syntactical format.
    - Flexibility: can be used with different types of data and with different levels of complexity.
    - Suitability: access various types of data and domain.

Parsers and Parsing

- XML Parsers
  - Parsers are software program that check the syntax used in an xml file.
  - Types of Parser:
    - Non-validating Parser:
      - Check the document follows the xml syntax rules.
      - It builds tree structure from the tags used in an XML document.
      - It is faster, because doesn’t check every element against DTD/Schema.
    - Validating Parser:
      - Checks the syntax, builds the tree structure.
      - Compares the structure of xml against DTD/Schema.

Contents

JAXP Interfaces

- JAXP (Java API for XML)
  - Is a collection of APIs that you can use in your Java applications to process and translate XML documents.
  - Consists of three APIs:
    - Simple API for XML (SAX): allows you to use a SAX parser to process the XML documents serially.
    - Document Object Model (DOM): allows you to use a DOM parser to process the XML documents in an object-oriented manner.
    - XML Stylesheet Language for Transformation (XSLT): allows you to transform XML documents in other formats, such as HTML.
  - Classifies the XML parser as:
    - Event-based Parsers: generate events while traversing through an XML document (SAX).
    - Object-based Parsers: arrange XML document in a tree-like structure (DOM).

JAXP Interfaces

- JAXP API packages

Packages	Descriptions
java.xml	Provides important XML constants and functionality from the XML specifications of W3C.
javax.xml.datatype	Provides mapping from XML to Java.
javax.xml.namespace	Provides classes for processing the XML namespace.
javax.xml.parsers	Provides classes for processing the XML documents.
javax.xml.transform	Defines the generic APIs for processing transformation instructions. Performing a transformation from source to result.

JAXP Interfaces

- JAXP API packages (cont)

Packages	Descriptions
javax.xml.transform.dom	Implements DOM specific transformation APIs.
javax.xml.transform.sax	Implements SAX2 specific transformation APIs.
javax.xml.transform.stream	Implements stream and URI specific transformation APIs.
javax.xml.validation	Provides an API for processing the XML documents.
javax.xml.xpath	Provides an object model neutral API for the evaluation of Xpath expressions. Access to the evaluation environment.

JAXP Interfaces

- Simple API for XML (SAX)
  - Is an event-based parser.
  - SAX parsers generate events and call event-handler in the application whenever it encounters elements or nodes in the document.
  - Is highly efficient in parsing large documents and requires less memory resources as compared to DOM to perform parsing.
  - The SAX API contains various classes containing methods to parse an XML document, such as *SAXParserFactory* and *SAXReader*.

JAXP Interfaces

- Simple API for XML (SAX) (cont)
  - Provide the classes and interfaces falling into five groups as following:

Groups	Descriptions
Interfaces Implemented by the Parser	Parser, Attribute List and Locator.
Interfaces Implemented by the Application	DocumentHandler, ErrorHandler, DTDHandler, EntityResolver.
Standard SAX classes	InputSource, SAXException, SAXParseException, HandlerBase.
Optional Java-specific Helper classes	ParserFactory, AttributeListImpl and LocatorImpl.
Java Demonstration classes	SystemIdDemo, ByteStreamDemo, CharacterStreamDemo, EntityDemo and DemoHandler.

JAXP Interfaces

- Simple API for XML (SAX) (cont)

Interfaces	Descriptions
Parser	Is used to register users for handling callbacks, which are the programs that respond to event. Triggers XML parsing and sets the locale for error reporting.
AttributeList	Allows users to traverse an attribute list. The parser may implement it in the same where the SAX driver is present or in a different class.
Locator	Allows users to find the current location in the XML source document. Has the same functions as in AttributeList interface.
DocumentHandler	Notifies the basic document-related events, such as start and end of elements.
ErrorHandler	Is implemented for error handling.
DTDHandler	Is implemented to receive notification of the NOTATION and ENTITY declarations.
EntityResolver	Is implemented to redirect the URIs in a document.

JAXP Interfaces

- Simple API for XML (SAX) (cont)

Classes	Descriptions
InputSource	Consists of all the necessary information for a single input source, such as public identifier, system identifier, byte stream, and character stream. Is instantiated by the application for the parser and others may be instantiated by the EntityHandler.
SAXException	Presents generalized SAX exceptions.
SAXParseException	Presents a SAX exception for a specific point in an XML source document.
HandleBase	Presents default implementations for DocumentHandler, ErrorHandler, DTDHandler and EntityResolver.
ParseFactor	Loads the SAX parser dynamically at run time depending on the class name.
AttributeListImpl	Generates a recurring copy of the AttributeList interface or it can also be used to supply a default implementation.
LocatorImpl	Makes a recurring snapshot of a locator's value at a specific point during parsing.

JAXP Interfaces

- Document Object Model (DOM)
  - Provides reference of the complete tree structure of an XML document and stores it in memory.
  - DOM parsers load the entire data into memory and arrange it in tree-like structure.
  - Uses various classes and interfaces that are defined in the DOM API to process XML document in the form a tree structure, known as DOM tree.
  - DOM tree consists of nodes, where each node represents a component of the XML document. Different types of nodes supported by DOM API are:
    - Element node.
    - Text node.
    - Attribute node.

JAXP Interfaces

- Document Object Model (DOM) (cont)

Classes	Descriptions
DocumentBuilderFactory	public abstract class DocumentBuilderFactory extends java.lang.Object Defines a factory API for enabling applications to get a parser for producing DOM object trees from XML documents Ex *DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance*
DocumentBuilder	public abstract class DocumentBuilder extends Object Defines the API to obtain DOM document instances from an XML documents Ex *DocumentBuilder domBuilder = documentFactory.newDocumentBuilder();*

JAXP Interfaces

- Document Object Model (DOM) (cont)

Interfaces	Descriptions
Document	public abstract interface Document extends Node Represents the entire XML document.
Element	public abstract interface Element extends Node Represents an element in an XML document.
Node	public abstract interface Node Provides the primary data type for the entire document structure.
NodeList	public abstract interface NodeList Presents a sequenced collection of nodes without defining or constraining the implementation of its collection.
ProcessingInstruction	public abstract interface ProcessingInstruction extends Node Stores processor-specific information in the text of the document.
Entity	public abstract interface Entity extends Node Presents an entity, either parsed or unparsed in an XML document.

JAXP Interfaces

- SAX vs. DOM

SAX	DOM
Random access to a document is prohibited since SAX processes the document sequentially.	Random access to a document is possible since DOM builds an in-memory image of the entire document.
Complex searches are not easy to perform.	Complex searches are easy to perform.
Data Type Definition (DTD) is not available.	DTD is available.
Lexical information is not available.	Lexical information is available.
SAX is read-only.	DOM can read, modify, search, add to, and delete from a document.
SAX is not supported by any browser.	DOM is supported by any browser.

JAXP Interfaces

- SAX vs. DOM (cont)
  - Benefits

SAX	DOM
SAX is able to parse the large files.	DOM is able to parse small files.
SAX is useful in retrieving a small subset of information.	DOM is useful in retrieving a subset of data.
SAX is faster as compared to DOM.	DOM is slower than SAX.

JAXP Interfaces

- XML Stylesheet Language for Transformation (XSLT)
  - Used to transform an XML document into other format such as HTML.
  - XSLT API need three components:
    - An instance of the TransformerFactory.
    - An instance of the Transformer.
    - The predefined transformation instructions.
  - The process of transformation:
    - This is done by transforming each XML element into an HTML element.
    - The XML source tree is transformed into result tree.
    - In the process of this transformation, XSLT uses Xpath that defines parts of the source document, which matches with a predefined template.
    - On finding out the match, XSLT transforms the matching part of the source document into the result document.

JAXP Interfaces

- Transformation API for XML (TrAX)

Classes	Descriptions
OutputKeys	public class OutputKeys extends java.lang.Object Presents string constraints, which are used to set output properties for a Transformer, or retrieve from a Transformer or Templates object.
Transformer	public abstract class Transformer extends Object Is used to transforming a source tree to a result tree.
TransformerFactory	public abstract class TransformerFactory extends java.lang.Object Creates instances for a Transformer or Templates object.

JAXP Interfaces

- Transformation API for XML (TrAX) (cont)

Interfaces	Descriptions
ErrorListener	public abstract interface ErrorListener Provides customization for error handling and uses the setErrorListener() method for registering an instance of the implementation with the Transformer.
Result	public abstract interface Result Implements an object, which contains the information required to build a transformation result tree.
Source	public abstract interface Source Implements an object, which contains the information required to act as source input.
SourceLocator	public abstract interface SourceLocator Reports an error that would have occurred in the source document or transformation instructions.
Templates	public abstract interface Templates An object, implementing this interface is the runtime representation of processed transformation instructions.
URIResolver	public abstract interface URIResolver Is called by the processor to turn a URI used in document(), xsl:import, or xsl:include into a source object.

Summary

- Overview of XML.
- Parsers and Parsing.
- JAXP Interfaces.