Lecture 12
Java XML & JSON Programming

presentation
Java Programming – Software App Development
Cristian Toma

D.I.C.E/D.E.I.C – Department of Economic Informatics & Cybernetics
www.dice.ase.ro

Cristian Toma – Business Card

Cristian Toma

IT&C Security Master

Dorobantilor Ave., No. 15-17
010572 Bucharest - Romania
http://ism.ase.ro
cristian.toma@ie.ase.ro
T +40 21 319 19 00 - 310
F +40 21 319 19 00

Agenda for Lecture 12

1 XML + XSD + XPath / XSLT

2 Java XML & JSON Programming

3 Exchange Ideas

1

XML & JSON Concepts

XML, XSD, X-Path / XSLT

1. **XML** Concepts

XML Concepts:

XML stands for e**X**tensible **M**arkup **L**anguage.

XML is designed to transport and store data.

XML is important to know, and very easy to learn.

Start learning XML now!

XML Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to> Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

XML Concepts:

What is XML?

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to carry data, not to display data
- XML tags are not predefined. You must define your own tags
- XML is designed to be self-descriptive
- XML is a W3C Recommendation

The Difference Between XML and HTML

XML is not a replacement for HTML.

XML and HTML were designed with different goals:

- XML was designed to transport and store data, with focus on what data is
- HTML was designed to display data, with focus on how data looks

HTML is about displaying information, while XML is about carrying information.

1. **XML** Concepts

XML Does Not DO Anything

Maybe it is a little hard to understand, but XML does not DO anything. XML was created to structure, store, and transport information.

The following example is a note to Tove, from Jani, stored as XML:

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The note above is quite self descriptive. It has sender and receiver information, it also has a heading and a message body.

But still, this XML document does not DO anything. It is just information wrapped in tags. Someone must write a piece of software to send, receive or display it.

With XML You Invent Your Own Tags

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

That is because the XML language has no predefined tags.

The tags used in HTML are predefined. HTML documents can only use tags defined in the HTML standard (like <p>, <h1>, etc.).

XML allows the author to define his/her own tags and his/her own document structure.

XML Concepts – What about JSON?:

XML is Not a Replacement for HTML

XML is a complement to HTML.

It is important to understand that XML is not a replacement for HTML. In most web applications, XML is used to transport data, while HTML is used to format and display the data.

My best description of XML is this:

XML is a software- and hardware-independent tool for carrying information.

XML is a W3C Recommendation

XML became a W3C Recommendation on February 10, 1998.

XML is Everywhere

XML is now as important for the Web as HTML was to the foundation of the Web.

XML is the most common tool for data transmissions between all sorts of applications.

XML Concepts:

XML Documents Form a Tree Structure

XML documents must contain a **root element**. This element is "the parent" of all other elements.

The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree.

All elements can have sub elements (child elements):

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

The terms parent, child, and sibling are used to describe the relationships between elements. Parent elements have children. Children on the same level are called siblings (brothers or sisters).

All elements can have text content and attributes (just like in HTML).

XML Concepts:

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```
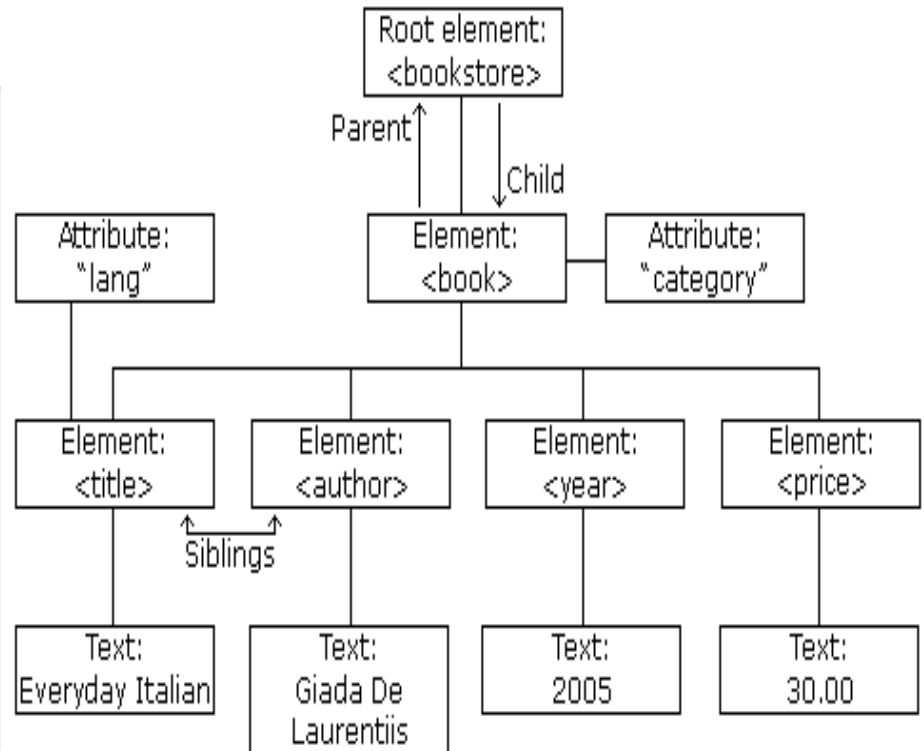
The root element in the example is <bookstore>. All <book> elements in the document are contained within <bookstore>.

The <book> element has 4 children: <title>,< author>, <year>, <price>.

1. **XML** Concepts

XML Concepts – Synthax:

All XML Elements Must Have a Closing Tag

In XML, it is illegal to omit the closing tag. All elements **must** have a closing tag:

<p>This is a paragraph.</p>

XML Tags are Case Sensitive

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.
Opening and closing tags must be written with the same case:

<Message>This is incorrect</message>
<message>This is correct</message>

Note: "Opening and closing tags" are often referred to as "Start and end tags"

1. **XML** Concepts

XML Concepts – Synthax:

XML Elements Must be Properly Nested
In XML, all elements **must** be properly nested within each other:

<i>This text is bold and italic</i>

In the example above, "Properly nested" simply means that since the <i> element is opened inside the <b> element, it must be closed inside the <b> element.

XML Documents Must Have a Root Element
XML documents must contain one element that is the **parent** of all other elements. This element is called the **root** element.

<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>

XML Concepts – Synthax:

XML Attribute Values Must be Quoted

XML elements can have attributes in name/value pairs just like in HTML.

In XML, the attribute values must always be quoted.

Study the two XML documents below. The first one is incorrect, the second is correct:

```
<note date=12/11/2007>
  <to>Tove</to>
  <from>Jani</from>
</note>


<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

The error in the first document is that the date attribute in the note element is not quoted.

1. **XML** Concepts

XML Concepts – Synthax:

Entity References
Some characters have a special meaning in XML.
If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:
<message>if salary < 1000 then</message>

To avoid this error, replace the "<" character with an **entity reference**:
<message>if salary < 1000 then</message>

There are 5 predefined entity references in XML:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Comments in XML
The syntax for writing comments in XML is similar to that of HTML.
<!-- This is a comment -->

XML Stores New Line as LF
* Windows applications store a new line as: carriage return and line feed (CR+LF).
* Linux/Unix and Mac OSX uses LF.

1. XML Concepts

What is an XML Element?

An XML element is everything from (including) the element's start tag to (including) the element's end tag. An element can contain: * other elements, * text, * attributes, * or a mix of all of the previous…

```
<bookstore>
 <book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
 </book>
 <book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
 </book>
</bookstore>
```

In the example above, <bookstore> and <book> have **element contents**, because they contain other elements. <book> also has an **attribute** (category="CHILDREN"). <title>, <author>, <year>, and <price> have **text content** because they contain text.

XML Concepts - Namespace:

Name Conflicts

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

This XML carries HTML table information:

```
<table>
 <tr>
  <td>Apples</td>
  <td>Bananas</td>
 </tr>
</table>
```

This XML carries information about a table (a piece of furniture):

```
<table>
 <name>African Coffee Table</name>
 <width>80</width>
 <length>120</length>
</table>
```

If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning.

A user or an XML application will not know how to handle these differences.

XML Concepts - Namespaces:

XML Namespaces - The xmlns Attribute
When using prefixes in XML, a so-called **namespace** for the prefix must be defined. The namespace is defined by the **xmlns attribute** in the start tag of an element. The namespace declaration has the following syntax. xmlns:*prefix*="*URI*". In the example above, the xmlns attribute in the <table> tag give the h: and f: prefixes a qualified namespace.
When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace. Namespaces can be declared in the elements where they are used or in the XML root element.

Note: The namespace URI is not used by the parser to look up information.
The purpose is to give the namespace a unique name. However, often companies use the namespace as a pointer to a web page containing namespace information.
Try to go to http://www.w3.org/TR/html4/

```
<root
xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/fur
niture">

<h:table>
 <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
 </h:tr>
</h:table>

<f:table>
 <f:name>African Coffee Table</f:name>
 <f:width>80</f:width>
 <f:length>120</f:length>
</f:table>

</root>
```

1. XML Concepts

Validating XML – XSD – XML Schema:

An XML Schema describes the structure of an XML document, just like a DTD.
An XML document with correct syntax is called "Well Formed".
An XML document validated against an XML Schema is both "Well Formed" and "Valid".

XML Schema

```
<xs:element name="note">

<xs:complexType>
  <xs:sequence>
   <xs:element name="to" type="xs:string"/>
   <xs:element name="from" type="xs:string"/>
   <xs:element name="heading" type="xs:string"/>
   <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

The XML Schema is interpreted like this:
- `<xs:element name="note">` defines the element called "note"
- `<xs:complexType>` the "note" element is a complex type
- `<xs:sequence>` the complex type is a sequence of elements
- `<xs:element name="to" type="xs:string">` the element "to" is of type string (text)
- `<xs:element name="from" type="xs:string">` the element "from" is of type string
- `<xs:element name="heading" type="xs:string">` the element "heading" is of type string
- `<xs:element name="body" type="xs:string">` the element "body" is of type string

Everything is wrapped in "Well Formed" XML.

1. **XML** Concepts

XML Schema:

Why Use an XML Schema?
With XML Schema, your XML files can carry a description of its own format.
With XML Schema, independent groups of people can agree on a standard for interchanging data.
With XML Schema, you can verify data

XML Schemas are More Powerful than DTD
XML Schemas are written in XML
XML Schemas are extensible to additions
XML Schemas support data types
XML Schemas support namespaces

XML Schemas Support Data Types
One of the greatest strength of XML Schemas is the support for data types:
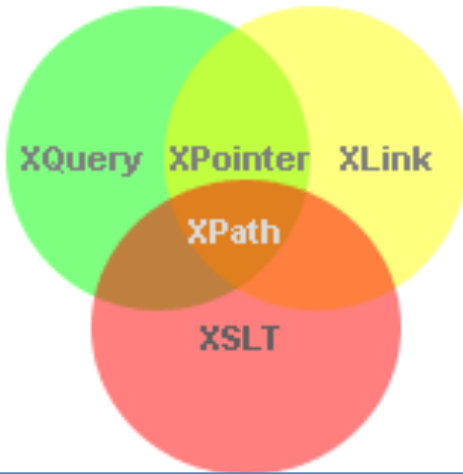It is easier to describe document content
It is easier to define restrictions on data
It is easier to validate the correctness of data
It is easier to convert data between different data types

1. XML Concepts

XML Path: is a language for finding information in an XML document.

- XPath is a syntax for defining parts of an XML document
- XPath uses path expressions to navigate in XML documents
- XPath contains a library of standard functions
- XPath is a major element in XSLT
- XPath is also used in XQuery, XPointer and XLink
- XPath is a W3C recommendation

XPath Path Expressions

XPath uses path expressions to select nodes or node-sets in an XML document. These path expressions look very much like the expressions you see when you work with a traditional computer file system.

Today **XPath** expressions can also be used in **JavaScript, Java, XML Schema, PHP, Python, C and C++**, and lots of other languages

XPath is Used in XSLT

XPath is a major element in the XSLT standard. Without XPath knowledge you will not be able to create XSLT documents.

Xpath + XSLT Transformations + JSON should be detailed

Section Conclusion

Fact: **Java is working with various data structures encoding**

In few **samples** it is simple to understand: XML and JSON.

2

Java XML – SAX, DOM, XSLT, XPath, JAXB 2, org.json

Java XML & JSON Programming

```java
package eu.ase.jaxb;

import java.util.ArrayList;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;

//This statement means that class "BookStore.java" is the root-element of our example
@XmlRootElement(namespace = "eu.ase.jaxb")
public class BookStore {

    // XmLElementWrapper generates a wrapper element around XML representation
    @XmlElementWrapper(name = "bookList")
    // XmlElement sets the name of the entities
    @XmlElement(name = "book")
    private ArrayList<Book> bookList;
    private String name;
    private String location;

    public void setBookList(ArrayList<Book> bookList) {
        this.bookList = bookList;
    }

    public ArrayList<Book> getBooksList() {
        return bookList;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getLocation() {
        return location;
    }
}
```

```java
import java.io.FileWriter;
import java.io.IOException;

public class BuildJson1 {
    public static void main(String[] args) {
        try {
            JSONObject dataset = new JSONObject();
            dataset.put("genre_id", 1);
            dataset.put("genre_parent_id", JSONObject.NULL);
            dataset.put("genre_title", "International");
            // use the accumulate function to add to an existing value. The value
            // will now be converted to a list
            dataset.accumulate("genre_title", "Pop");
            // append to the key
            dataset.append("genre_title", "slow");
            dataset.put("genre_handle", "International");
            dataset.put("genre_color", "#CC3300");
            // get the json array for a string
            System.out.println(dataset.getJSONArray("genre_title"));
            // prints ["International","Pop","slow"]
            // increment a number by 1
            dataset.increment("genre_id");
            // quote a string allowing the json to be delivered within html
            System.out.println(JSONObject.quote(dataset.toString()));
            System.out.println("\n\nWrite to the file:\n\n");
            System.out.println(dataset.toString());

            FileWriter fw = new FileWriter(new File("myJsonObj.json"));
            fw.write(dataset.toString());
            fw.close();

            // prints
            // "{\"genre_color\":\"#CC3300\",\"genre_title\":[\"International\",\"Pop\",\"slow\"],
            // \"genre_handle\":\"International\",\"genre_parent_id\":null,\"genre_id\":2}"
        } catch (JSONException jsone) {
                jsone.printStackTrace();
        } catch (IOException ioe) {
                ioe.printStackTrace();
        }
    } //end main
} //end class
```

Section Conclusions

Please review Java XML and JSON samples.

Java XML-JSON Programming
for easy sharing

3

Java Programming & XML | JSON

Communicate & Exchange Ideas

Questions & Answers!

But wait...
There's More!

Thanks!

Java SE
End of Lecture 12 – XML | JSON Programming