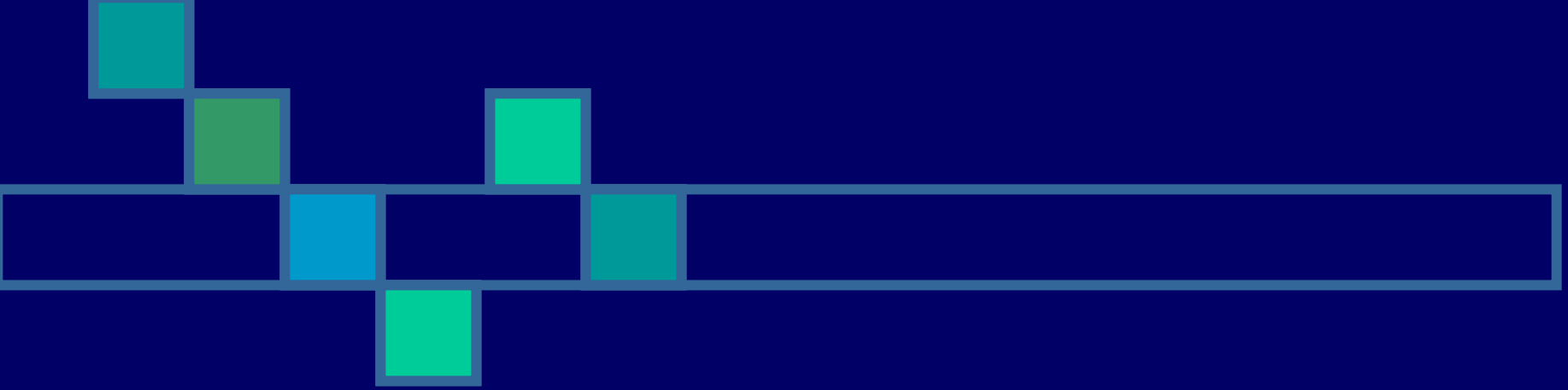

XML

Technical Talk

by Svetlana Slavova

Outline

- Introduction to XML
- XML vs. Serialization
- Curious facts, advantages & weaknesses
- XML syntax
- Parsing XML
- Example
- References

Introduction to XML (I)

- XML (E**X**tensible **M**arkup **L**anguage) is a standard
- Mechanism for encoding data that is independent of any programming language
- XML is used for data exchange

Introduction to XML (II)

- XML has a nested structure that allows to describe hierarchical data sets
- XML describes the meaning of data, not how to display them (HTML describes how to display the data)
- The recipient can easily parse the XML document & get the necessary data

XML vs. Serialization

- Object serialization can be used to exchange binary data that describe the objects
- In particular, the recipient must be a Java/C# program that uses the same versions of classes as the sending program

XML – curious facts ☺

- XML was designed by an 11-member group that never met face-to-face
- The design was accomplished using weekly email teleconferences
- The major design decisions were reached only in 20 weeks in 1996
- XML 1.0 became a W3C Recommendation in 1998

XML – advantages

- Human- & machine-readable format
- Can represent the most general computer science data structures: records, lists, and trees
- The strict syntax allows the parsing algorithms to remain simple, efficient, and consistent

XML – weaknesses

- The parsers have to check if the XML documents are properly formatted
- The basic parsing requirements do not support a wide variety of data types => additional work is necessary in order to process the desired data from a XML document

XML syntax (I)

- XML structure

XML declaration

XML schema

XML elements

XML syntax (II)

- XML declaration

<?xml version="1.0" encoding="UTF-8"?>

- XML schema

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

XML syntax (III)

- XML elements

<element_tag> content </element_tag>
<element_tag attributes> content </ element_tag>
<element_tag attributes />

<temperature>-30</temperature>
<temperature name="C">-30</temperature>
<temperature value="-30" />

XML syntax (IV)

```
<coin value="1" name="loonie" />
```

```
<coin>
   <name>loonie</name>
   <value>1</value>
</coin>
```

XML syntax (IV)

- Character encoding – Unicode
  <俄语>Данни</俄语>

XML syntax (V)

- Every XML document must have exactly one top-level root element

```
<?xml version="1.0" encoding="UTF-8"?>
<class842>CMPT 842</class842>
<class880>CMPT 880</class880>
```

Wrong!

```
<?xml version="1.0" encoding="UTF-8"?>
<classes>
    <class842>CMPT 842</class842>
    <class880>CMPT 880</class880>
</classes>
```

Correct!

Parsing XML

- SAX (Simple Access to XML) Parser
  - Event-driven parser – whenever the parser encounters a particular construct, then it calls a method that you must provide
  - More efficient for large documents – do not build a tree
  - More complex
- DOM (Document Object Model) Parser
  - Builds up a tree that represents the XML document
  - Easy to use

Example (I)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
   1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
   transitional.dtd">
<coin>
   <name>loonie</name>
   <value>1</value>
</coin>
```

Example (II)

```
/*
 * Parser.java
 * @author Svetlana Slavova, sds797@mail.usask.ca
 * @version 1.0, February, 2006
 */

package xmlparsing;

import java.util.Vector;
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
```

```
public class Parser
{
   //To get a DOM document from the XML document
    private DocumentBuilder builder;

    /** Creates a new instance of Parser. */
    public Parser()
    {
        try
        {
            //The program obtains a parser that creates the DOM tree
            DocumentBuilderFactory factory =
                                        DocumentBuilderFactory.newInstance();
            builder = factory.newDocumentBuilder();
        }
        catch (Exception ex)
        {
            System.out.println("Error in the constructor: " + ex);
        }
    }
```

```
/** Parses the XML document. */
 public Vector parse(String filename)
 {
     try
     {
         File f = new File(filename);
         //Create the DOM tree
         Document doc = builder.parse(f);
         //Get the root element of the tree
         Element root = doc.getDocumentElement();
         Vector v = getCoin(root);
         return v;
     }
     catch (Exception ex)
     {
         System.out.println("Error in method parse: " + ex);
         return null;
     }
 }
```

```
/** Gets information about the coin. */
private Vector getCoin(Element e)
{
    Vector items = new Vector();

    int value = -1;
    String name = "";

    try
    {
        //Get the children
        NodeList children = e.getChildNodes();

        for (int i = 0; i < children.getLength(); i++)
        {
            //Get a node
            Node childNode = children.item(i);
```

```
if (childNode instanceof Element)
{
    Element childElement = (Element) childNode;
    String tagname = childElement.getTagName();

    Text textNode = (Text) childElement.getFirstChild();
    String data = textNode.getData();

    if (tagname.equals("value"))
        value = Integer.parseInt(data);
    else
    {
        if (tagname.equals("name"))
            name = data;
        else
            System.out.println("Unknown tag element");
    }
}
} //End for loop
```

```
            //Add the name & the value to the Vector
            items.addElement(name);
            items.addElement(value);

            return items;

        } //End try block
        catch (Exception ex)
        {
            System.out.println("Error in method
                                        getCoin: " + ex);
            return null;
        }
    } //End getCoin method
```

```
    public static void main(String[] args)
    {
        Parser p = new Parser();

        String filename = "../XML.xml";

        Vector items = p.parse(filename);

        System.out.println("Coin information:");
        System.out.println("name = " + items.elementAt(0));
        System.out.println("value = " + items.elementAt(1));
        System.out.println("--------------------------------");
    }

} //End of the class
```

Example (III)

- Result

Coin information:

name = loonie

value = 1

References

- Big Java, Cay Horstmann
- http://en.wikipedia.org/wiki/XML