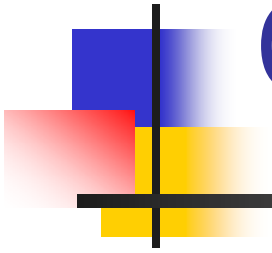
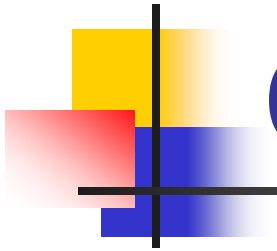


COMP4317: XML & Database

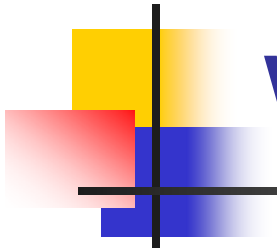
Tutorial 1 – Week 2

Thang Bui @ CSE.UNSW



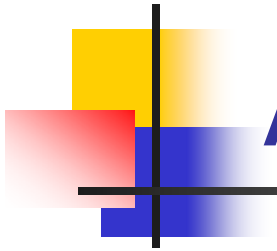
Content

- Why tutorial?
- XML
- DOM
- Java & DOM
- C++ & DOM
- Discussion



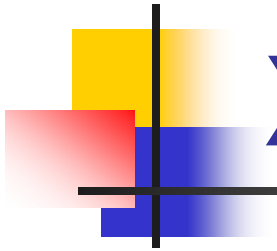
Why tutorial

- Discussing about the assignments
- Talking about (possible) open lecture issues



Assignments

- Totally 5 assignments
- Programming issues:
  - Platforms:
    - Linux (CSE machine)
  - Programming languages:
    - C/C++: gcc/g++
    - Java: javac
  - Tools:
    - Xerces: http://xerces.apache.org



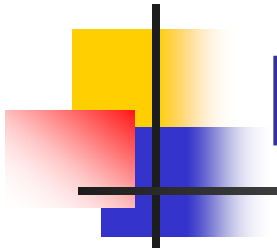
XML Introduction

- XML ?
  - stands for eXtensible Markup Language
  - to describe data
  - define your own tags
  - uses Document Type Definition (DTD) or XML Scheme to describe data structure
  - is a W3C Recommendation since 1998

XML example

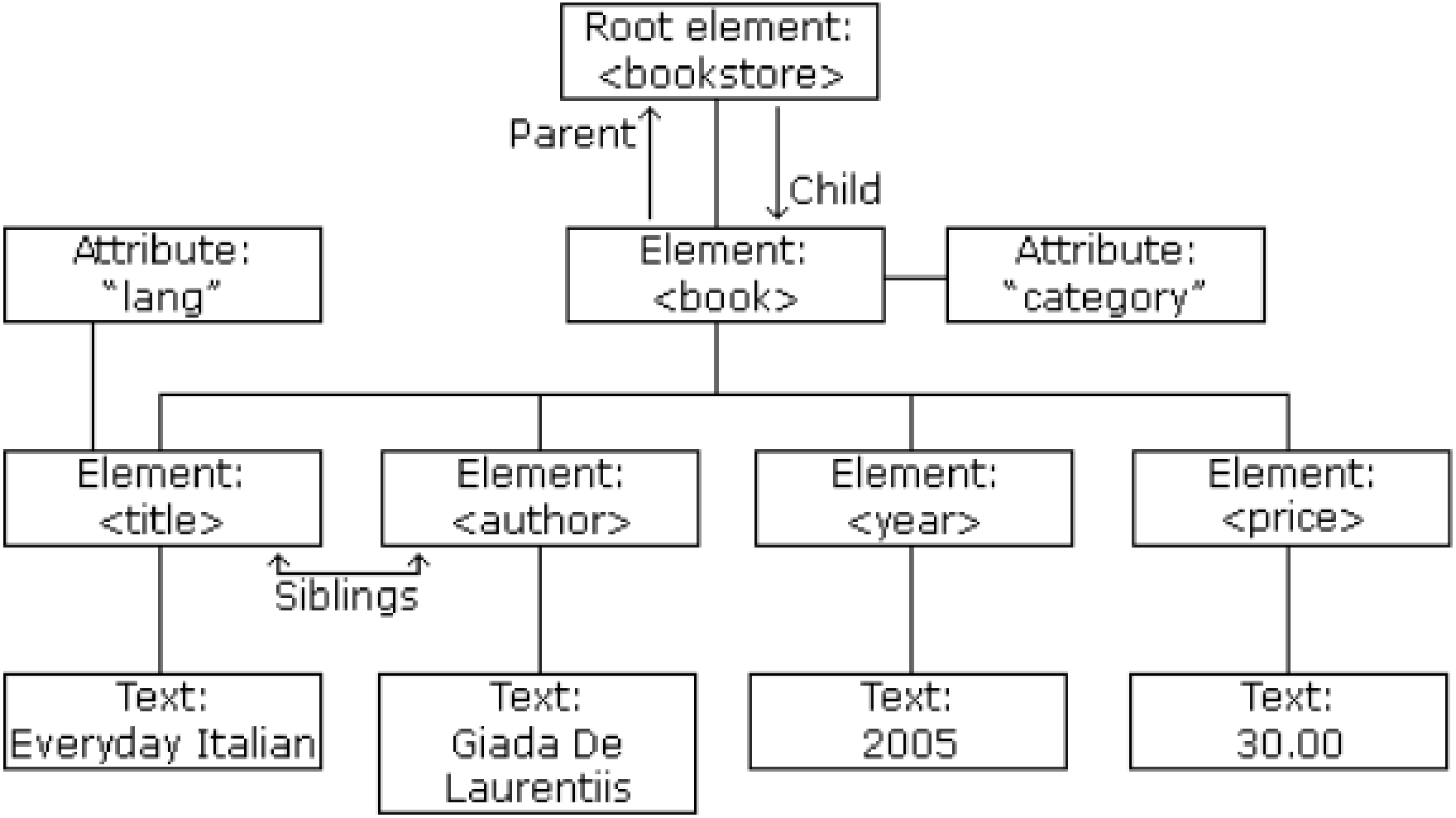
```
<library>
  <book isbn="1-2345-6789-0" year="1994">
    <title>TCP/IP Illustrated</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price currency="USD">65.95</price>
  </book>
  <book isbn="0-9876-5432-1" year="1999">
    <title>The Economics of Technology and Content for Digital TV</title>
    <editor><last>Gerbarg</last><first>Darcy</first></editor>
    <publisher>Kluwer Academic Publishers</publisher>
    <price currency="USD">129.95</price>
  </book>
</library>
```

DOM – Document Object Model

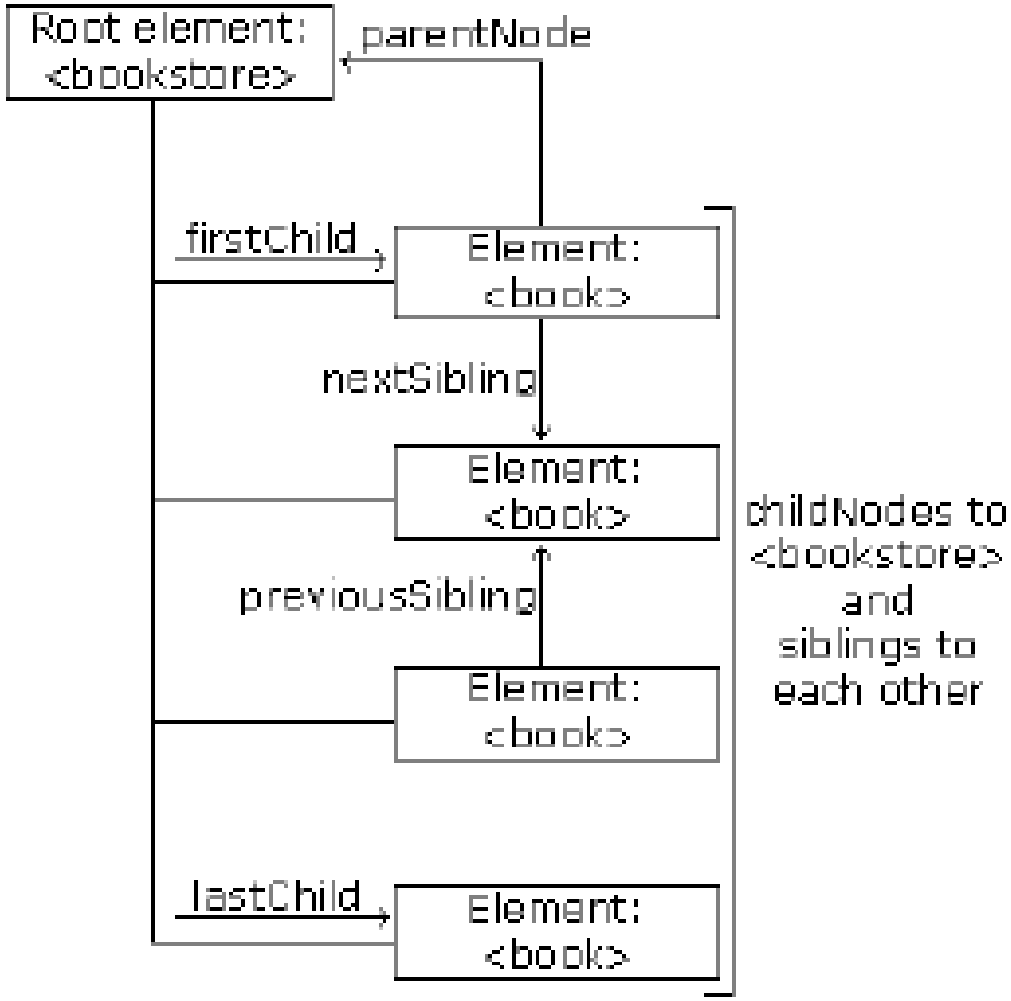


- is a W3C standard .
- platform- and language-independent.
- defines a standard set of objects for XML.
- defines a standard way to access/manipulate XML documents.
- views XML documents as a tree-structure. All elements/text/ attributes can be accessed/manipulate (as nodes) through the DOM tree.

DOM Tree Example



DOM Tree Navigation



Parsing: XML Resource => DOM Tree

Java code

```
import javax.xml.parsers.*;
import org.w3c.dom.*;
class XMLTester {
  XMLTester() {
    try {
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder docBuilder = factory.newDocumentBuilder();
      Document doc = docBuilder.parse("input.xml");
      System.out.println(doc);
    } catch(Exception e) { e.printStackTrace(); }
  }
 public static void main(String[] args) {
    XMLTester tester = new XMLTester();
  }
}
```

Parsing: XML Resource => DOM Tree

Java code

```
import javax.xml.parsers.*;
import org.w3c.dom.*;
class XMLTester {
  XMLTester() {
    try {
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder docBuilder = factory.newDocumentBuilder();
      Document doc = docBuilder.parse("input.xml");
      System.out.println(doc);
    } catch(Exception e) { e.printStackTrace(); }
  }
 public static void main(String[] args) {
    XMLTester tester = new XMLTester();
  }
}
```

DOM Tree as a Document Node

Parsing: XML Resource => DOM Tree

Java code

```
import javax.xml.parsers.*;
import org.w3c.dom.*;
class XMLTester {
  XMLTester() {
    try {
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder docBuilder = factory.newDocumentBuilder();
      Document doc = docBuilder.parse("input.xml");
      System.out.println(doc);
    } catch(Exception e) { e.printStackTrace(); }
  }
 public static void main(String[] args) {
    XMLTester tester = new XMLTester();
  }
}
```

[#document: null]

Why?

Parsing: XML Resource => DOM Tree

C++ code

```
#include <iostream.h>
#include <xercesc/util/PlatformUtils.hpp>
#include <xercesc/dom/DOM.hpp>
#include <xercesc/util/XMLUniDefs.hpp>
XERCES_CPP_NAMESPACE_USE

int main() {
  XMLPlatformUtils::Initialize();
  static const XMLCh gLS[]={chLatin_L, chLatin_S, chNull};
  DOMImplementation *impl=DOMImplementationRegistry::getDOMImplementation(gLS);
  DOMBuilder * parser = ((DOMImplementationLS*)impl)->createDOMBuilder
                                    (DOMImplementationLS::MODE_SYNCHRONOUS, 0);
  DOMDocument * doc = parser->parseURI("input.xml");
  cout << doc << endl;
  XMLPlatformUtils::Terminate();
}
```

DOM Tree as a Document Node

Parsing: XML Resource => DOM Tree

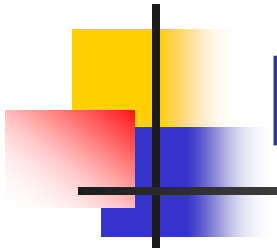
C++ code

```
#include <iostream.h>
#include <xercesc/util/PlatformUtils.hpp>
#include <xercesc/dom/DOM.hpp>
#include <xercesc/util/XMLUniDefs.hpp>
XERCES_CPP_NAMESPACE_USE

int main() {
  XMLPlatformUtils::Initialize();
  static const XMLCh gLS[]={chLatin_L, chLatin_S, chNull};
  DOMImplementation *impl=DOMImplementationRegistry::getDOMImplementation(gLS);
  DOMBuilder * parser = ((DOMImplementationLS*)impl)->createDOMBuilder
                                    (DOMImplementationLS::MODE_SYNCHRONOUS, 0);
  DOMDocument * doc = parser->parseURI("input.xml");
  cout << doc << endl;
  XMLPlatformUtils::Terminate();
}
```

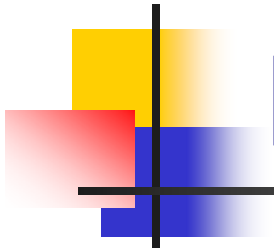
0x8054c08

Why?



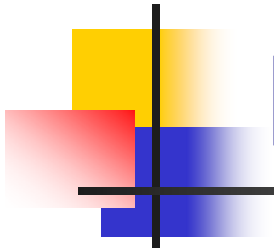
DOM Data Structure in Java

- Node
  - Primary DOM data type for all data in XML DOM Tree: Document, Element, Attribute, Text, ...
  - Inherited classes:
    - Document
    - Element
    - Attr
    - ...



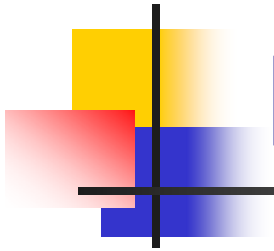
DOM Data Structure in Java

- Node
  - Type of the node: getNodeType()
  - Children: getChildNodes(), getFirstChild(), getLastChild()
  - Sibling: getNextSibling(), getPreviousSibling()
  - Attributes (of Element): getAttributes()
  - ...



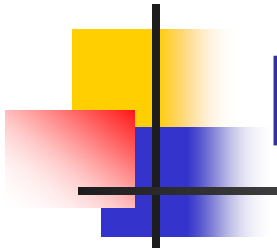
DOM Data Structure in Java

- NodeList
  - An order list of nodes
  - Return by getChildNode(), ...
  - Only two methods:
    - getLength()
    - item(int i)



DOM Data Structure in Java

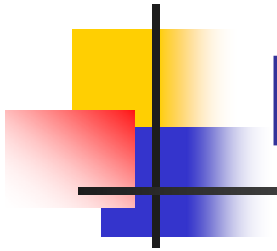
- NamedNodeMap
  - Collection of nodes
  - Return by getAttributes(), ...
  - Can access by names (of nodes)
  - Few methods:
    - getLength()
    - item(int i)
    - ...



DOM Data Structure in Java

Reversing Document Nodes

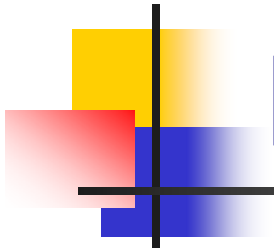
```
public void reverse(Document doc) {
  Stack stack = new Stack();
  Element root = doc.getDocumentElement();
  NodeList elements = root.getChildNodes();
  while(elements.getLength() > 0) {
    stack.push(elements.item(0));
    root.removeChild(elements.item(0));
  }
  while(stack.size() > 0) {
    root.appendChild((Node) stack.pop());
  }
}
```



DOM Data Structure in Java

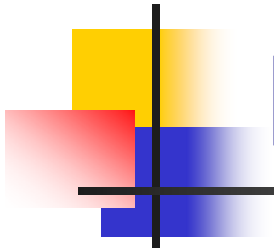
Travelling Element and Text Nodes

```
public void travel(Node n) {
  switch (n.getNodeType()) {
    case Node.ELEMENT_NODE:
      System.out.print("Node: " + n.getNodeName());
      Node child = n.getFirstChild();
      while (child != null) {
        travel(child); child = child.getNextSibling();
      }
      break;
    case Node.TEXT_NODE:
      System.out.print("Text: " + n.getNodeValue()); break;
  }
}
```



DOM Data Structure in C++

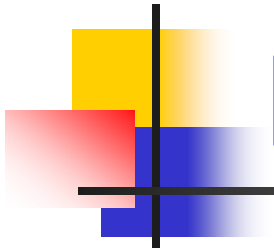
- DOMNode
  - DOMDocument
  - DOMElement
  - DOMAttr
  - ...
- DOMNodeList
- DOMNamedNodeMap



DOM Data Structure in C++

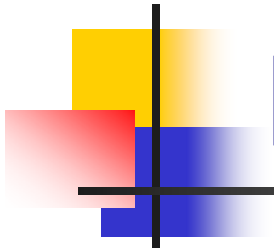
Count attributes of a node

```
int count_attr(DOMNode * node) {
  int num = 0, i;
  DOMNamedNodeMap * attrs = node->getAttributes();
  for (i = 0; i < attrs->getLength(); i++) {
    DOMNode * attr = attrs->item (i);
    if (attr->getNodeType() == DOMNode::ATTTRIBUTE_NODE)
      num++;
  }
  return num;
}
```



DOM Text Data in C++

- Character: XMLCh ≠ char
- XMLString class
  - static char * transcode (const XMLCh * const toTranscode)
  - static XMLCh * transcode (const char * const toTranscode)



DOM Text Data in C++

Print attributes of a node

```
void print_attr(DOMNode * node) {
  int i;
  DOMNamedNodeMap * attrs = node->getAttributes();
  for (i = 0; i < attrs->getLength(); i++) {
    DOMNode * attr = attrs->item (i);
    char * name = XMLString::transcode(attr->getNodeName());
    char * val = XMLString::transcode(attr->getNodeValue());
    cout << name << "=\"" << val << "\"" << endl;
    XMLString::release(&name);
    XMLString::release(&val);
  }
}
```

DOM Text Data in C++

Print attributes of a node

```
void print_attr(DOMNode * node) {
  int i;
  DOMNamedNodeMap * attrs = node->getAttributes();
  for (i = 0; i < attrs->getLength(); i++) {
    DOMNode * attr = attrs->item (i);
    char * name = XMLString::transcode(attr->getNodeName());
    char * val = XMLString::transcode(attr->getNodeValue());
    cout << name << "=\"" << val << "\"" << endl;
    XMLString::release(&name);
    XMLString::release(&val);
  }
}
```

isbn="1-2345-6789-0"
year="1994"

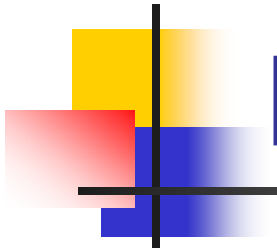
DOM Text Data in C++

Print attributes of a node

```
void print_attr(DOMNode * node) {
  int i;
  DOMNamedNodeMap * attrs = node->getAttributes();
  for (i = 0; i < attrs->getLength(); i++) {
    DOMNode * attr = attrs->item (i);
    char * name = XMLString::transcode(attr->getNodeName());
    char * val = XMLString::transcode(attr->getNodeValue());
    cout << name << "=\"" << val << "\"" << endl;
    XMLString::release(&name);
    XMLString::release(&val);
  }
}
```

From XMLCh * to char *

Release yourself!



DOM Text Data in C++

- Making a utility class for Text datatype in C++ => why not?
  - Constructor: XMLCh* -> char *
  - Destructor: release
  - Operator<<: to print out using operator "<<"
  - Do it youself!

DOM Text Data in C++

Print attributes of a node

```
void print_attr(DOMNode * node) {
  int i;
  DOMNamedNodeMap * attrs = node->getAttributes();
  for (i = 0; i < attrs->getLength(); i++) {
    DOMNode * attr = attrs->item (i);
    cout << myXMLStr(attr->getNodeName())
         << "=\"" << myXMLStr(attr->getNodeValue()) << "\""
         << endl;
  }
}
```

Constructor: transcode (to local “char *”)

Operator<<: return the local “char *”

Destructor: release the local “char *”

DOMParser: Easier Way ?

Java code

```
import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.*;
class XMLTester {
  XMLTester() {
    try {
      DOMParser theParser = new DOMParser();
      theParser.parse("input.xml");
      Document doc = theParser.getDocument();
      System.out.println(doc);
    } catch(Exception e) { e.printStackTrace(); }
  }
 public static void main(String[] args) {
    XMLTester tester = new XMLTester();
  }
}
```

Need “xerces.jar” in the classpath

DOMParser: Easier Way ?

C++ code

```
#include <iostream.h>
#include <xercesc/util/PlatformUtils.hpp>
#include <xercesc/parsers/XercesDOMParser.hpp>
XERCES_CPP_NAMESPACE_USE

int main() {
  XMLPlatformUtils::Initialize();
  XercesDOMParser * theParser = new XercesDOMParser();

  theParser->parse("input.xml");
  DOMDocument *doc = theParser->getDocument();
  cout << doc << endl;
  XMLPlatformUtils::Terminate();
}
```