



www.sbVB.com.br

XML and systems integration

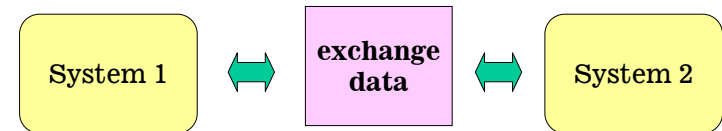
- Version 8.0 of March 10th, 2006
- By Sergio Barbosa Villas-Boas
 - www.sbv.com.br
 - sbvb@sbvb.com.br



www.sbVB.com.br

Integration of 2 software systems

- Often there's need to integrate 2 software systems. For instance: the system of a book distributor and the system of a book retail shop.



2



www.sbVB.com.br

XML: data and metadata

- The data should be saved with its metadata. So, just by reading XML one should know about what the data is about.



www.sbVB.com.br

Positional files

```
06022006      nome do produto33278      antonio
05022006      doce de jaboticaba15502      ze
```

3



www.sbVB.com.br

Tokenized files

```
06;02;2006;nome do produto;3327;8;antonio
05;02;2006;doce de jaboticaba;1550;2;ze
```



www.sbVB.com.br

SOME C++ LIBRARIES FOR XML (available with source)

- **Xerces – XML parser/validator SAX, DOM**
<http://xml.apache.org/xerces-c/index.html>
- **Xalan – XSLT processor**
<http://xml.apache.org/xalan-c/index.html>
Apache Software Licence (can be used commercially)
- **Expat – XML parser SAX**
<http://expat.sourceforge.net/>
- **libxml – XML parser**
<http://xmlsoft.org/>

6



www.sbVB.com.br

Some libraries for XML (that can be used from C++)

- **xml4c (by IBM, not free)**
<http://alphaworks.ibm.com/tech/xml4c>
- **MSXML (by MS, Windows only)**
<http://microsoft.com/xml/>
<http://msxml.com/>

7



www.sbVB.com.br

HTML: displays the information

```
<h1>Bibliography</h1>
<p>
<i>Microelectronics Circuits</i><br>
Sedra / Smith<br>
HRW Saunders, 1991
<p>
<i>Practical Unix Security</i><br>
Simson Garfinkel e Gene Spafford<br>
O´Reilly, 1994
```

Bibliography

Microelectronics Circuits
Sedra / Smith
HRW Saunders, 1991

Practical Unix Security
Simson Garfinkel e Gene Spafford
O'Reilly, 1994

8



www.sbVB.com.br

XML: DESCRIBES THE content (data and metadata)

```

<?xml version="1.0"?>
<!-- bib.xml -->
<!-- data, metadata -->
<bibliography>
  <book>
    <title>Microelectronics Circuits</title>
    <author>Sedra / Smith</author>
    <editor>HRW Saunders</editor>
    <year>1991</year>
  </book>
  <book>
    <title>Practical Unix Security</title>
    <author>Simson Garfinkel</author>
    <editor>O'Reilly</editor>
    <year>1994</year>
  </book>
</bibliography>

```



www.sbVB.com.br

XML: IS unicode here a sample of a xml file in Japanese

```

<?xml version="1.0"?>
<!-- bib-jap.xml -->
<文献>
  <本>
    <タイトル>日本語の基礎</タイトル>
    <著者>Sedra / Smith</著者>
    <出版社>HRW Saunders</出版社>
    <年>1991</年>
  </本>
  <本>
    <タイトル>Practical Unix Security</タイトル>
    <著者>Simson Garfinkel</著者>
    <出版社>O'Reilly</出版社>
    <年>1994</年>
  </本>
</文献>

```



www.sbVB.com.br

Database

• Database categories

- Hierarchic
- Network
- Relational (uses PK, FK and SQL)
- Object oriented

Examples:

- Oracle
- MS SQL server
- DB2
- Sybase
- Informix
- MySQL
- PostgreSQL (*)

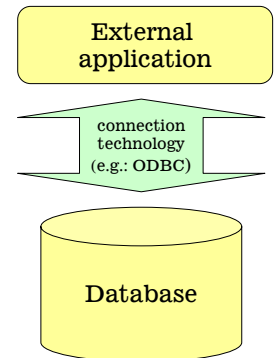
(*) free



www.sbVB.com.br

Database (2)

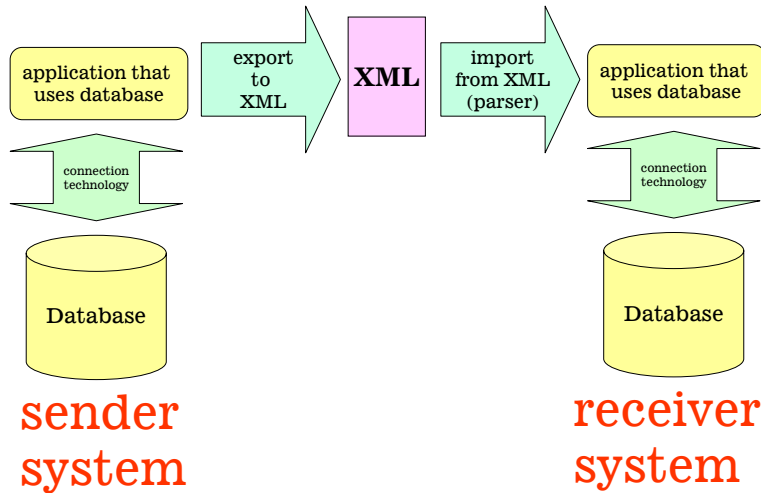
- A database is a software that creates a tier over the file system, to provide a "data server".
- The database is developed in a way that external applications can be developed to use it.
- The external application can be developed using any computer language.
- There should be a technology to connect the external application to the database e.g: ODBC. Other example: SQLAPI++ (a good alternative for those who use C++).





www.sbVB.com.br

Integrating 2 systems using XML

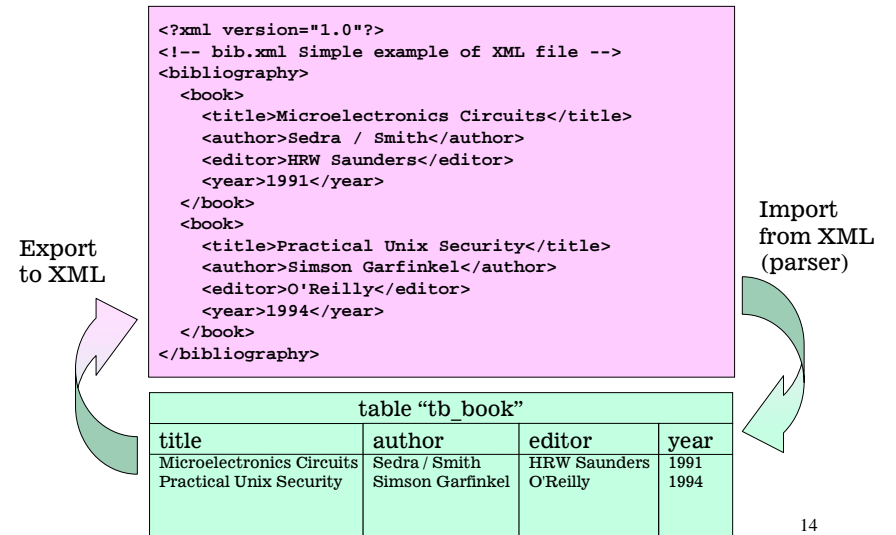


13



www.sbVB.com.br

Integrating 2 systems using XML (2)



14



www.sbVB.com.br

Creating table book in MySQL

```

mysql -u root -p
<root password>
create database db_sbvb;
grant all on db_sbvb.* to sbvb identified by 'pwd';
connect db_sbvb;
create table tb_book (title varchar(30), author varchar(30),
  editor varchar(30), year int(4));
insert into tb_book (title,author,editor,year)
  values ("Microelectronics Circuits","Sedra / Smith","HRW Saunders",1991);
insert into tb_book (title,author,editor,year)
  values ("Practical Unix Security","Simson Garfinkel","O'Reilly",1994);
show tables;
+-----+
| Tables_in_db_sbvb |
+-----+
| tb_book            |
+-----+
select * from tb_book;
+-----+-----+-----+-----+
| title                | author          | editor          | year |
+-----+-----+-----+-----+
| Microelectronics Circuits | Sedra / Smith  | HRW Saunders   | 1991 |
| Practical Unix Security  | Simson Garfinkel | O'Reilly       | 1994 |
+-----+-----+-----+-----+

```



www.sbVB.com.br

Creating table book in PostgreSQL

```

CREATE TABLE tb_book ( author text, editor text, title text,
  year int4);

insert into tb_book (title,author,editor,year)
  values ('Microelectronics Circuits','Sedra / Smith',
  'HRW Saunders',1991);

insert into tb_book (title,author,editor,year)
  values ('Practical Unix Security','Simson Garfinkel',
  'O'Reilly',1994);

```

| author text | editor text | title text | year int4 |
|--------------------|--------------|---------------------------|-----------|
| 1 Sedra / Smith | HRW Saunders | Microelectronics Circuits | 1991 |
| 2 Simson Garfinkel | O'Reilly | Practical Unix Security | 1994 |

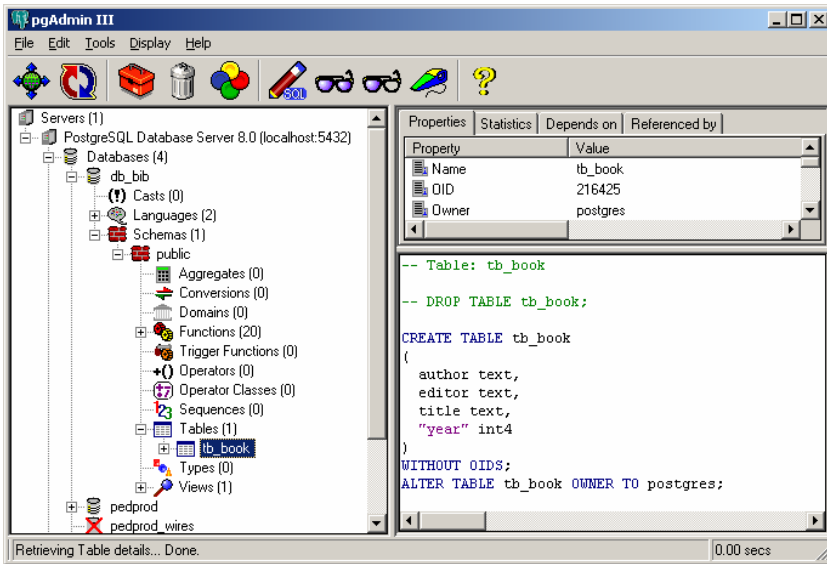
2 rows.

16



www.sbVB.com.br

Creating table book in PostgreSQL (2)



www.sbVB.com.br

Connect C++ to database using library SQLAPI++

```
#include <iostream>
#include <SQLAPI.h> // main SQLAPI++ header
int main()
{
  using namespace std;
  SAConnection con; // create connection object
  try
  {
    // con.Connect(<database name>,<user name>,<password>,<client>);
    con.Connect("<computerName>@db_sbvb","sbvb","pwd", SA_MySQL_Client);
    cout << "We are connected!" << endl;
    con.Disconnect();
    cout << "We are disconnected!" << endl;
  }
  catch(SAException & x)
  {
    // print error message
    cout << (const char*)x.ErrText() << endl;
  }
  return 0;
}
```

Explanation why connection did not have success

We are connected!
We are disconnected!

Login failed for user 'testLogin'.



www.sbVB.com.br

Export data to XML

```
#include <iostream>
#include <SQLAPI.h>
using namespace std;
void outTable(SAConnection & con, ostream & out = cout)
{
  SAString command;
  command = "select * from tb_book"; // a SQL command
  SACommand cmd(&con, command); // create command object
  cmd.Execute(); // Select from our test table
  SAString title, author, editor, year;
  out << "<?xml version='1.0'?>" << endl;
  out << "<bibliography>" << endl;
  while(cmd.FetchNext()) {
    out << "<book>" << endl;
    title = (const char *)cmd.Field("title").asString();
    author = (const char *)cmd.Field("author").asString();
    editor = (const char *)cmd.Field("editor").asString();
    year = (const char *)cmd.Field("year").asString();
    out << "<title>" << (const char *)title << "</title>" << endl;
    out << "<author>" << (const char *)author << "</author>" << endl;
    out << "<editor>" << (const char *)editor << "</editor>" << endl;
    out << "<year>" << (const char *)year << "</year>" << endl;
    out << "</book>" << endl;
  }
  out << "</bibliography>" << endl;
}
```



www.sbVB.com.br

Export data to XML (2)

```
int main()
{
  SAConnection con; // connection object
  try
  {
    con.Connect("<computerName>@db_sbvb","sbvb","pwd", SA_MySQL_Client);
    cout << "Connection ok" << endl;

    outTable(con);

    con.Disconnect();
    cout << "We are disconnected!" << endl;
  }
  catch(SAException &x)
  {
    cout << "Exception: " << (const char*)x.ErrText() << endl;
    return -1;
  }
  return 0;
}
```

Output is the same as the XML file



www.sbVB.com.br

Export data to XML, v2

```
#include <iostream>
#include <deque>
#include "vblib.h"
#include <SQLAPI.h>
using namespace std;
typedef deque<VBString> StringContainer;
void outTable2(SAConnection & con, StringContainer & sc, ostream & out = cout)
{
    StringContainer::iterator it = sc.begin();
    SAString command;
    command = "select * from ";
    command += *it++; // first string of sc is the table name
    // increment the iterator to point to xml root tag
    VBString xmlRootTag = *it++;
    // increment the iterator to point to xml leave
    VBString xmlLeave = *it++;
    // increment the iterator to point to column names
    SACommand cmd(&con, command); // create command object
    cmd.Execute(); // Select from our test table
    SAString title, author, editor, year;
    out << "<?xml version='1.0'?" << endl;
    out << "<" << xmlRootTag << ">" << endl;
    VBString column;
    StringContainer::iterator it_save = it;
    // save the iterator in the beginning of the column names
```



www.sbVB.com.br

Export data to XML, v2 (2)

```
while(cmd.FetchNext())
{
    out << "<" << xmlLeave << ">" << endl;
    it = it_save;
    // get the iterator in the beginning of the column names
    for (it = it_save; it != sc.end() ; it++)
    {
        column = *it;
        out << "<" << column << ">" <<
            (const char *)cmd.Field(column.c_str()).asString() <<
            "</" << column << ">" << endl;
    }
    out << "</" << xmlLeave << ">" << endl;
}
out << "</" << xmlRootTag << ">" << endl;
}
```



www.sbVB.com.br

Export data to XML, v2 (3)

```
int main()
{
    SAConnection con; // connection object
    try
    {
        con.Connect("<computerName>@db_sbvb","sbvb","pwd", SA_MySQL_Client);
        cout << "Connection ok" << endl;

        StringContainer sc;
        sc.push_back("book_tb");
        sc.push_back("bibliography");
        sc.push_back("book");
        sc.push_back("title");
        sc.push_back("author");
        sc.push_back("editor");
        sc.push_back("year");
        outTable2(con,sc);

        con.Disconnect();
        cout << "We are disconnected!" << endl;
    }
    catch(SAException &x)
    {
        cout << "Exception: " << (const char*)x.ErrText() << endl;
    }
    return 0;
}
```

Output is the same as the XML file



www.sbVB.com.br

XML definitions

- Tags or elements:
 - <book>...</book>, <author>...</author>
 - Begin tag: <book> , end tag : </book>
 - Empty elements: <year></year>, or for short: <year/>
 - There must exist only one root element
 - The elements are nested in a tree structure
- The XML file is a text file
 - XML files should support unicode format, for tag names and values.
- Attributes of a tag:
 - Ex.: <book língua="Português" preço="R\$100.00"/>



XML modeling languages

- Validating a XML
 - DTD: Document Type Definition (the most traditional)
 - XML Schema (MS, W3C)
- 2 Boolean attributes of a given XML data
 - well formed: true or false.*
 - is it a xml or not?*
 - This attribute depends only on the data itself, and not on the model of the data. If the XML data is well formed, its possible to parse the data.
 - valid: true or false.*
 - For a given data model , a given well formed XML data is valid or not valid. The data model is represented using XML modeling languages.



XML data well formed and not well formed

```
<?xml version="1.0"?>
<!-- bib2.xml -->
<bibliography>
  <book title="Microelectronics Circuits"
        author="Sedra / Smith" editor="HRW Saunders"
        year="1991"/>
  <book title="Practical Unix Security"
        author="Simson Garfinkel" editor="O'Reilly"
        year="1994"/>
</bibliography>
```

well formed true

```
<?xml version="1.0"?>
<!-- bib3.xml -->
<bibliography>
  <book1></book2>
</bibliography>
```

well formed false



XML validation

- There are 2 main languages to describe validation of XML: DTD and Schema.
- DTD is more traditional.
- Schema has some more features, and is a XML describing a XML.



XML validation with DTD

```
<?xml version="1.0"?>
<!-- bib.xml -->
<!DOCTYPE bibliography SYSTEM "bib.dtd">
<bibliography>
  <book>
    <title>Microeletronics Circuits</title>
    <author>Sedra / Smith</author>
    <editor>HRW Saunders</editor>
    <year>1991</year>
  </book>
  <book>
    <title>Practical Unix Security</title>
    <author>Simson Garfinkel</author>
    <editor>O'Reilly</editor>
    <year>1994</year>
  </book>
</bibliography>
```

XML

```
<!-- bib.dtd -->
<!ELEMENT bibliography (book*)>
<!ELEMENT book (title,author,editor,year)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT editor (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```

DTD

Don't forget Space here



DTD extern to XML data

```

<!-- t1.dtd -->
<!ELEMENT name (entry_1*, entry_2?)>
<!ELEMENT entry_1 (#PCDATA)>
<!ELEMENT entry_2 (#PCDATA)>
<!-- #PCDATA = Parsed Character Data -->

```

| element | multiplicity |
|---------|------------------|
| (blank) | 1 (1..1) |
| + | 1 or more (1..*) |
| * | 0 or more (0..*) |
| ? | 0 or 1 (0..1) |

```

<?xml version="1.0"?>
<!-- t1.xml -->
<!DOCTYPE name SYSTEM "t1.dtd">
<name>
  <entry_1>xyz</entry_1>
  <entry_1>xyz</entry_1>
  <entry_1>xyz</entry_1>
  <entry_2>xyz</entry_2>
</name>

```



DTD inside XML data

```

<?xml version="1.0"?>
<!-- t3.xml -->
<!DOCTYPE name [
  <!ELEMENT name (entry_1*, entry_2?)>
  <!ELEMENT entry_1 (#PCDATA)>
  <!ELEMENT entry_2 (#PCDATA)>
]>
<name>
  <entry_1>xyz</entry_1>
  <entry_1>xyz</entry_1>
  <entry_1>xyz</entry_1>
  <entry_2>xyz</entry_2>
</name>

```



DTD blocks

- Elements
 - Ex: "book", "person"
- Tags
 - Ex: <book> <person>
- Attributes
 - Ex: <book author="sbVB">
- Entities: Variables that are changed to text
 - "<" to "<"
 - ">" to ">"
 - "&" to "&"
 - """ to "
 - "'" to '
- PCDATA: Parsed Character Data
- CDATA: Character Data (not parsed)



DTD elements

• Declaring Element

```
<!ELEMENT element-name (element-content)>
```

• Empty elements

```
<!ELEMENT element-name (EMPTY)>
```

```
Ex: <!ELEMENT img (EMPTY)>
```

• Elements with data

```
<!ELEMENT element-name (#CDATA)>
```

```
<!ELEMENT element-name (#PCDATA)>
```

```
<!ELEMENT element-name (ANY)>
```

```
Ex: <!ELEMENT note (#PCDATA)>
```

#CDATA means the element contains character data that is not supposed to be parsed by a parser.

#PCDATA means that the element contains data that IS going to be parsed by a parser.

The keyword ANY declares an element with any content.

If a #PCDATA section contains elements, these elements must also be declared.



DTD elements (2)

Elements with child

```
<!ELEMENT element-name (child-element-name)>
<!ELEMENT element-name (child_1,child_2,...)>
```

When children are declared in a sequence separated by commas, the children must appear in the same sequence in the document. In a full declaration, the children must also be declared, and the children can also have children. The full declaration of the note document will be:

Ex:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#CDATA)>
<!ELEMENT from (#CDATA)>
<!ELEMENT heading (#CDATA)>
<!ELEMENT body (#CDATA)>
```



DTD attributes

• Declaring attributes

```
<!ATTLIST element-name attribute-name attribute-type default-value>
```

• Possible attribute types

- CDATA: char data (string)
- (eval|eval|..): enumerate
- ID: unique identification
- IDREF: id of an element
- IDREFS: a list of ids
- NMTOKEN: a valid XML name
- NMTOKENS: O valor é uma lista de nomes XML válidos
- ENTITY: O valor é uma entidade
- ENTITIES: O valor é uma lista de entidades
- NOTATION: O valor é o nome de uma notation
- xml: O valor é pré definido

• Possible default values

- #DEFAULT value: Valor implícito (default) do atributo
- #REQUIRED: O atributo precisa existir no elemento
- #IMPLIED: O atributo não precisa existir no elemento
- #FIXED value: O valor do atributo é fixo



XML with DTD intern and attributes

```
<?xml version="1.0"?>
<!-- t2.xml -->
<!DOCTYPE db [
<!ELEMENT db (person*,book*)>
<!ELEMENT person (name)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT book (title)>
<!ELEMENT title (#PCDATA)>
<!ATTLIST book author CDATA #REQUIRED ling CDATA #REQUIRED>
]>
<db>
  <person id="o1">
    <name>Alan</name>
  </person>
  <person id="o2">
    <name>Rafael</name>
  </person>
  <book author="Sr. author" ling="Port">
    <title>book de XML</title>
  </book>
</db>
```



XML with DTD intern and attributes (2)

```
<?xml version="1.0"?>
<!-- t4.xml -->
<!DOCTYPE geometry [
<!ELEMENT geometry (square*)>
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
]>
<geometry>
  <square width="100"/>
</geometry>
```



www.sbVB.com.br

XML validation with Schema

// todo

```
<?xml version="1.0"?>
<!-- bib.xml -->
<bibliography>
  <book>
    <title>Microeletronics Circuits</title>
    <author>Sedra / Smith</author>
    <editor>HRW Saunders</editor>
    <year>1991</year>
  </book>
  <book>
    <title>Practical Unix Security</title>
    <author>Simson Garfinkel</author>
    <editor>O'Reilly</editor>
    <year>1994</year>
  </book>
</bibliography>
```

37



www.sbVB.com.br

XML parser: SAX or DOM?

- **DOM (Document Object Model):** object driven strategy. First get all XML data, and then navigate through the structure.
 - Easier to use. All XML data is loaded to memory, so it is not recommended for big XML files.
- **SAX (Simple Api for XML):** event driven strategy. Hook user methods to be called during parse.
 - Never have all XML data in memory. Recommended when the XML file is big.

38



www.sbVB.com.br

XML parser: SAX or DOM? (2)

- If your XML file is small, DOM is usually the best choice
- If your XML file is large, the usage of DOM still works, but the consumption of memory might become a problem; that memory problem usually turns into a low performance problem.
 - If the data model of the XML file is complex, it is hard to solve the problem
 - If the data model of the XML file is simple, and the reason for the largeness of the file is that there are many registers, you can solve that using SAX

39



www.sbVB.com.br

SAX XML Parse with Xerces C++

- Get binary file xerces-c-src_2_6_0.zip, and open it to a working directory.
- // todo

40



www.sbVB.com.br

DOM XML Parse with Xerces C++ (Win / VC version)

- Get binary file xerces-c-src_2_6_0.zip, and open it to a working directory.

41



www.sbVB.com.br

DOM XML Parse with Xerces C++ (linux / g++ version)

- Get binary file xerces-c-src_2_6_0.tar.gz, and open it to a working directory

```
tar -zxvf xerces-c-src_2_6_0.tar.gz
```
- Go to directory xerces-c-src_2_6_0/src/xercesc
- Create environment variable with the command below

```
export XERCESROOT=$HOME/xerces-c-src_2_6_0
```
- Execute command “./configure -prefix=\$HOME”. If your computer has g++ properly installed, that will be done with success
- Execute command “make && make install”. That will start the compilation of the library. It takes a while to complete.
 - If you have 2 processors, add “-j2” in the make command. That will compile with both cpu’s.
- To compile myxml.cpp, use the command below

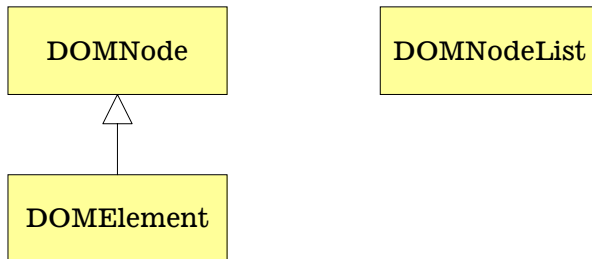
```
g++ -I$HOME/lib -I$HOME/include -lxerces-c -lpthread myxml.cpp -o myxml
```

42



www.sbVB.com.br

DOM parser



www.sbVB.com.br

DOM XML Parse with Xerces Java

- Get binary file Xerces-J-bin.2.6.2.zip, and open it to a working directory. For example C:\winap\xerces-2_6_2_Java. Let’s parse the same xml file, as below.

```

<?xml version="1.0"?>
<bibliography>
  <book>
    <title>Microelectronics Circuits</title>
    <author>Sedra / Smith</author>
    <editor>HRW Saunders</editor>
    <year>1991</year>
  </book>
  <book>
    <title>Practical Unix Security</title>
    <author>Simson Garfinkel</author>
    <editor>O'Reilly</editor>
    <year>1994</year>
  </book>
</bibliography>
  
```

44



www.sbVB.com.br

DOM XML Parse with Xerces Java

(2)

```

package br.com.sbVB.VBXML;

import java.io.PrintWriter;
import org.w3c.dom.Attr;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import dom.Counter;
import dom.GetElementsByTagName;
import dom.ParserWrapper;

public class VBDomParse {
    protected static final String NAMESPACES_FEATURE_ID =
        "http://xml.org/sax/features/namespaces";
    protected static final String VALIDATION_FEATURE_ID =
        "http://xml.org/sax/features/validation";
    protected static final String SCHEMA_VALIDATION_FEATURE_ID =
        "http://apache.org/xml/features/validation/schema";
    protected static final String SCHEMA_FULL_CHECKING_FEATURE_ID =
        "http://apache.org/xml/features/validation/schema-full-checking";
    protected static final String DYNAMIC_VALIDATION_FEATURE_ID =
        "http://apache.org/xml/features/validation/dynamic";

```

45



www.sbVB.com.br

DOM XML Parse with Xerces Java

(3)

```

protected static final String DEFAULT_PARSER_NAME =
    "dom.wrappers.Xerces";
protected static final String DEFAULT_ELEMENT_NAME = "";
protected static final boolean DEFAULT_NAMESPACES = true;
protected static final boolean DEFAULT_VALIDATION = false;
protected static final boolean DEFAULT_SCHEMA_VALIDATION = false;
protected static final boolean DEFAULT_SCHEMA_FULL_CHECKING = false;
protected static final boolean DEFAULT_DYNAMIC_VALIDATION = false;

public static void main(String argv[]) {
    String fileToParse = "bib.xml";
    Counter counter = new Counter();
    PrintWriter out = new PrintWriter(System.out);
    ParserWrapper parser = null;
    String elementName = DEFAULT_ELEMENT_NAME;
    String attributeName = null;
    boolean namespaces = DEFAULT_NAMESPACES;
    boolean validation = DEFAULT_VALIDATION;
    boolean schemaValidation = DEFAULT_SCHEMA_VALIDATION;
    boolean schemaFullChecking = DEFAULT_SCHEMA_FULL_CHECKING;
    boolean dynamicValidation = DEFAULT_DYNAMIC_VALIDATION;

```



www.sbVB.com.br

DOM XML Parse with Xerces Java

(4)

```

// create parser
try {
    parser = (ParserWrapper)Class.forName(DEFAULT_PARSER_NAME).newInstance();
}
catch (Exception e) {
    System.err.println("error: Unable to instantiate parser ("
        +DEFAULT_PARSER_NAME+");");
}

// set parser features
try {
    parser.setFeature(NAMESPACES_FEATURE_ID, namespaces);
    parser.setFeature(VALIDATION_FEATURE_ID, validation);
    parser.setFeature(SCHEMA_VALIDATION_FEATURE_ID, schemaValidation);
    parser.setFeature(SCHEMA_FULL_CHECKING_FEATURE_ID, schemaFullChecking);
    parser.setFeature(DYNAMIC_VALIDATION_FEATURE_ID, dynamicValidation);
}
catch (SAXException e) {
    System.err.println("warning: Failed setting parser features");
}

```

47



www.sbVB.com.br

DOM XML Parse with Xerces Java

(5)

```

// parse file
try {
    Document document = parser.parse(fileToParse);
    Element rootElement = document.getDocumentElement(); // bibliography
    NodeList bookList = rootElement.getElementsByTagName("book");
    for (int i=0; i < bookList.getLength(); i++) {
        Element bookElement;
        if (bookList.item(i).getNodeType() == Element.ELEMENT_NODE) {
            bookElement = (Element)bookList.item(i);
            Node node = bookElement.getElementsByTagName("title").item(0);

            if (node.hasChildNodes()) {
                Node iteratorNode = node.getFirstChild();
                while (true) {
                    if (iteratorNode.getNodeType() == Element.TEXT_NODE){
                        String title = iteratorNode.getNodeValue();
                        System.out.println("Title="+title);
                    } // while(true)

                    // if this is the last node
                    if(iteratorNode == node.getLastChild()) break;

                    iteratorNode.getNextSibling(); // go for next node
                } // end of if has child nodes
            } // if Element.ELEMENT_NODE
        } // for
    } // try
}

```



DOM XML Parse with Xerces Java

(6)

```

catch (SAXParseException e)
{
    System.err.println("SAXParseException");
    e.printStackTrace(System.err);
}
catch (Exception e)
{
    System.err.println("Exception");
    e.printStackTrace(System.err);
}
} // end of main
} // end of class

```



DOM XML Parse with Xerces Java

(3)

```

// parse file
try {
    Document document = parser.parse(fileToParse);
    Element rootElement = document.getDocumentElement(); // bibliography
    NodeList bookList = rootElement.getElementsByTagName("book");
    for (int i=0; i < bookList.getLength(); i++)
    {
        // System.out.println("i="+i); // debug
        Element bookElement;
        if (bookList.item(i).getNodeType() == Element.ELEMENT_NODE)
        {
            // System.out.println("item="+i); // debug
            bookElement = (Element)bookList.item(i);
            Node node = bookElement.getElementByTagName("title").item(0);

            if (node.hasChildNodes())
            {
                Node iteratorNode = node.getFirstChild();
                while (true)
                {
                    // System.out.println("iteratorNode"); // debug
                    if (iteratorNode.getNodeType() == Element.TEXT_NODE)
                    {
                        String title = iteratorNode.getNodeValue();
                        System.out.println("title="+title);
                    }

                    // if this is the last node
                    if (iteratorNode == node.getLastChild())
                        break;

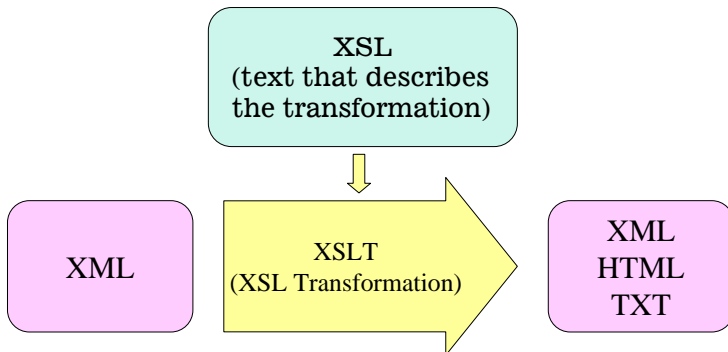
                    iteratorNode.getNextSibling(); // go for next node
                } // end of if has child nodes
            }
        }
    }
} catch (SAXParseException e) {
    System.err.println("SAXParseException");
    e.printStackTrace(System.err);
} catch (Exception e) {
    System.err.println("Exception");
    e.printStackTrace(System.err);
}
}

```



XSLT: the XSL transformation

- Good for “XML behind the scenes”

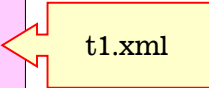


Example of XSLT: XML to HTML

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>title cd 1</title>
    <artist>artist cd 1</artist>
    <country>country cd 1</country>
    <company>company cd 1</company>
    <price>price cd 1</price>
    <year>year cd 1</year>
  </cd>
  <cd>
    <title>title cd 2</title>
    <artist>artist cd 2</artist>
    <country>country cd 2</country>
    <company>company cd 2</company>
    <price>price cd 2</price>
    <year>year cd 2</year>
  </cd>
  <cd>
    <title>title cd 3</title>
    <artist>artist cd 3</artist>
    <country>country cd 3</country>
    <company>company cd 3</company>
    <price>price cd 3</price>
    <year>year cd 3</year>
  </cd>
</catalog>

```





Example of XSLT: XML to HTML (2)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th align="left">Title</th>
<th align="left">Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

t1.xsl

Constants
In red



Example of XSLT XML to HTML (3)

```
xalan t1.xml t1.xsl
```

```
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th align="left">Title</th>
<th align="left">Artist</th>
</tr>
<tr>
<td>title cd 1</td><td>artist cd 1</td>
</tr>
<tr>
<td>title cd 2</td><td>artist cd 2</td>
</tr>
<tr>
<td>title cd 3</td><td>artist cd 3</td>
</tr>
</table>
</body>
</html>
```



XSLT (2)

```
<?xml version="1.0"?>
<!-- t5.xml -->
<bib>
<book>
<title>book 1</title>
<author>author 1</author>
</book>
<paper>
<title>Paper 1</title>
</paper>
<paper>
<title>book 2</title>
</paper>
<paper>
<title>Paper 2</title>
</paper>
<book>
<title>book 3</title>
<author>author 3</author>
</book>
</bib>
```

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999
/XSL/Transform">
<xsl:template match="/">
<xsl:for-each select="bib/**">
*)<xsl:value-of select="title"/>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8"?>
*)book 1
*)Paper 1
*)book 2
*)Paper 2
*)book 3
```



XSLT (3)

```
<?xml version="1.0"?>
<catalog>
<cd>
<title>Sampa</title>
<artist>Caetano Veloso</artist>
<country>Brazil</country>
<price>10.90</price>
<year>1985</year>
</cd>
<cd>
<title>Construção</title>
<artist>Chico Buarque</artist>
<country>Brazil</country>
<price>11.20</price>
<year>1988</year>
</cd>
<cd>
<title>Vamos Fugir</title>
<artist>Gilberto Gil</artist>
<country>Brazil</country>
<price>9.70</price>
<year>1989</year>
</cd>
</catalog>
```

t6.xml



XSLT (4)

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th>Title</th>
<th>Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<xsl:if test="price > 10">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="artist"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

t6.xml

price > 10



XSLT (5)

xalan t6.xml t6.xsl

```

<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th>Title</th><th>Artist</th>
</tr>
<tr>
<td>Sampa</td><td>Caetano Veloso</td>
</tr>
<tr>
<td>Construcao</td><td>Chico Buarque</td>
</tr>
</table>
</body>
</html>

```



XSLT (6)

```

<?xml version="1.0"?>
<catalog>
<cd>
<title>Sampa</title>
<artist>Caetano Veloso</artist>
<country>Brazil</country>
<price>10.90</price>
<year>1985</year>
</cd>
<cd>
<title>Construcao</title>
<artist>Chico Buarque</artist>
<country>Brazil</country>
<price>11.20</price>
<year>1988</year>
</cd>
<cd>
<title>Vamos Fugir</title>
<artist>Gilberto Gil</artist>
<country>Brazil</country>
<price>9.70</price>
<year>1989</year>
</cd>
<minVal>8</minVal>
</catalog>

```



XSLT (7)

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="minVal" select="catalog/minVal"/>
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th>Title</th>
<th>Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<xsl:if test="price > $minVal">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="artist"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

define variable minVal

price > minVal



www.sbVB.com.br

Real State (imovel)

61



www.sbVB.com.br

More information

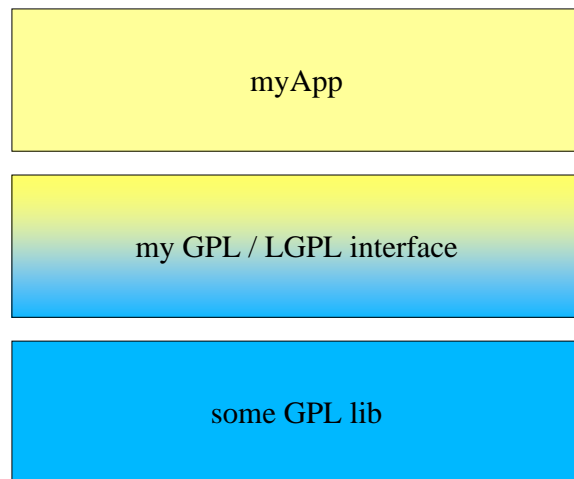
- <http://www.w3schools.com/xsl/>
- <http://www.cogx.com/>

62



www.sbVB.com.br

Solving the GPL problem



www.sbVB.com.br

危機