```
This file was updated on Saturday, 2012-10-13 at 3:13 PM


========================================================================
sample.xml
========================================================================


<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--
  File:   sample.xml
  Author: Jesse M. Heines, UMass Lowell CS, heines@cs.uml.edu

  Downloaded on October 13, 2012, 2:55 PM
  Source:  http://www.yolinux.com/TUTORIALS/XML-Xerces-C.html
-->
<root>
   <ApplicationSettings
           option_a = "10"
           option_b = "24"
           >
   </ApplicationSettings>
   <OtherStuff
           option_x = "500"
           >
   </OtherStuff>
</root>


========================================================================
parser.hpp
========================================================================


/*
 * File:   parser.hpp
 * Author: Jesse M. Heines, UMass Lowell CS, heines@cs.uml.edu
 *
 * Downloaded on October 13, 2012, 2:55 PM
 * Source:  http://www.yolinux.com/TUTORIALS/XML-Xerces-C.html
 */

#ifndef XML_PARSER_HPP
#define XML_PARSER_HPP
/**
 *  @file
 *  Class "GetConfig" provides the functions to read the XML data.
 *  @version 1.0
 */
#include <xercesc/dom/DOM.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMDocumentType.hpp>
#include <xercesc/dom/DOMElement.hpp>
#include <xercesc/dom/DOMImplementation.hpp>
#include <xercesc/dom/DOMImplementationLS.hpp>
#include <xercesc/dom/DOMNodeIterator.hpp>
#include <xercesc/dom/DOMNodeList.hpp>
#include <xercesc/dom/DOMText.hpp>

#include <xercesc/parsers/XercesDOMParser.hpp>
#include <xercesc/util/XMLUni.hpp>

#include <string>
#include <stdexcept>
```

```
 65
 66  // Error codes
 67
 68  enum {
 69     ERROR_ARGS = 1,
 70     ERROR_XERCES_INIT,
 71     ERROR_PARSE,
 72     ERROR_EMPTY_DOCUMENT
 73  };
 74
 75  class GetConfig
 76  {
 77  public:
 78     GetConfig();
 79    ~GetConfig();
 80     void readConfigFile(std::string&) throw(std::runtime_error);
 81
 82     char *getOptionA() { return m_OptionA; };
 83     char *getOptionB() { return m_OptionB; };
 84
 85  private:
 86     xercesc::XercesDOMParser *m_ConfigFileParser;
 87     char* m_OptionA;
 88     char* m_OptionB;
 89
 90     // Internal class use only. Hold Xerces data in UTF-16 SMLCh type.
 91
 92     XMLCh* TAG_root;
 93
 94     XMLCh* TAG_ApplicationSettings;
 95     XMLCh* ATTR_OptionA;
 96     XMLCh* ATTR_OptionB;
 97  };
 98  #endif
 99
100
101  ======================================================================
102  parser.cpp
103  ======================================================================
104
105
106  /*
107   * File:   parser.cpp
108   * Author: Jesse M. Heines, UMass Lowell CS, heines@cs.uml.edu
109   *
110   * Downloaded on October 13, 2012, 3:03 PM
111   * Source:  http://www.yolinux.com/TUTORIALS/XML-Xerces-C.html
112   */
113
114  #include <string>
115  #include <iostream>
116  #include <sstream>
117  #include <stdexcept>
118  #include <list>
119
120  #include <sys/types.h>
121  #include <sys/stat.h>
122  #include <unistd.h>
123  #include <errno.h>
124
125  #include "parser.hpp"
126
127  using namespace xercesc;
128  using namespace std;
```

```
129
130  /**
131   *  Constructor initializes xerces-C libraries.
132   *  The XML tags and attributes which we seek are defined.
133   *  The xerces-C DOM parser infrastructure is initialized.
134   */
135
136  GetConfig::GetConfig()
137  {
138     try
139     {
140        XMLPlatformUtils::Initialize();  // Initialize Xerces infrastructure
141     }
142     catch( XMLException& e )
143     {
144        char* message = XMLString::transcode( e.getMessage() );
145        cerr << "XML toolkit initialization error: " << message << endl;
146        XMLString::release( &message );
147        // throw exception here to return ERROR_XERCES_INIT
148     }
149
150     // Tags and attributes used in XML file.
151     // Can't call transcode till after Xerces Initialize()
152     TAG_root        = XMLString::transcode("root");
153     TAG_ApplicationSettings = XMLString::transcode("ApplicationSettings");
154     ATTR_OptionA = XMLString::transcode("option_a");
155     ATTR_OptionB = XMLString::transcode("option_b");
156
157     m_ConfigFileParser = new XercesDOMParser;
158  }
159
160  /**
161   *  Class destructor frees memory used to hold the XML tag and
162   *  attribute definitions. It als terminates use of the xerces-C
163   *  framework.
164   */
165
166  GetConfig::~GetConfig()
167  {
168     // Free memory
169
170     delete m_ConfigFileParser;
171     if(m_OptionA)   XMLString::release( &m_OptionA );
172     if(m_OptionB)   XMLString::release( &m_OptionB );
173
174     try
175     {
176        XMLString::release( &TAG_root );
177
178        XMLString::release( &TAG_ApplicationSettings );
179        XMLString::release( &ATTR_OptionA );
180        XMLString::release( &ATTR_OptionB );
181     }
182     catch( ... )
183     {
184        cerr << "Unknown exception encountered in TagNamesdtor" << endl;
185     }
```

```
186
187     // Terminate Xerces
188
189     try
190     {
191        XMLPlatformUtils::Terminate();  // Terminate after release of memory
192     }
193     catch( xercesc::XMLException& e )
194     {
195        char* message = xercesc::XMLString::transcode( e.getMessage() );
196
197        cerr << "XML ttolkit teardown error: " << message << endl;
198        XMLString::release( &message );
199     }
200  }
201
202  /**
203   *  This function:
204   *  - Tests the access and availability of the XML configuration file.
205   *  - Configures the xerces-c DOM parser.
206   *  - Reads and extracts the pertinent information from the XML config file.
207   *
208   *  @param in configFile The text string name of the HLA configuration file.
209   */
210
211  void GetConfig::readConfigFile(string& configFile)
212          throw( std::runtime_error )
213  {
214     // Test to see if the file is ok.
215
216     struct stat fileStatus;
217
218     int iretStat = stat(configFile.c_str(), &fileStatus);
219     if( iretStat == ENOENT )
220        throw ( std::runtime_error("Path file_name does not exist, or path is an empty string.") );
221     else if( iretStat == ENOTDIR )
222        throw ( std::runtime_error("A component of the path is not a directory."));
223     else if( iretStat == ELOOP )
224        throw ( std::runtime_error("Too many symbolic links encountered while traversing the path."));
225     else if( iretStat == EACCES )
226        throw ( std::runtime_error("Permission denied."));
227     else if( iretStat == ENAMETOOLONG )
228        throw ( std::runtime_error("File can not be read\n"));
229
230     // Configure DOM parser.
231
232     m_ConfigFileParser->setValidationScheme( XercesDOMParser::Val_Never );
233     m_ConfigFileParser->setDoNamespaces( false );
234     m_ConfigFileParser->setDoSchema( false );
235     m_ConfigFileParser->setLoadExternalDTD( false );
236
237     try
238     {
239        m_ConfigFileParser->parse( configFile.c_str() );
240
241        // no need to free this pointer - owned by the parent parser object
242        DOMDocument* xmlDoc = m_ConfigFileParser->getDocument();
243
244        // Get the top-level element: NAme is "root". No attributes for "root"
245
246        DOMElement* elementRoot = xmlDoc->getDocumentElement();
247        if( !elementRoot ) throw(std::runtime_error( "empty XML document" ));
248
```

```
249         // Parse XML file for tags of interest: "ApplicationSettings"
250         // Look one level nested within "root". (child of root)
251
252         DOMNodeList*      children = elementRoot->getChildNodes();
253         const  XMLSize_t nodeCount = children->getLength();
254
255         // For all nodes, children of "root" in the XML tree.
256
257         for( XMLSize_t xx = 0; xx < nodeCount; ++xx )
258         {
259            DOMNode* currentNode = children->item(xx);
260            if( currentNode->getNodeType() &&  // true is not NULL
261                currentNode->getNodeType() == DOMNode::ELEMENT_NODE ) // is element
262            {
263               // Found node which is an Element. Re-cast node as element
264               DOMElement* currentElement
265                           = dynamic_cast< xercesc::DOMElement* >( currentNode );
266               if( XMLString::equals(currentElement->getTagName(), TAG_ApplicationSettings))
267               {
268                  // Already tested node as type element and of name "ApplicationSettings".
269                  // Read attributes of element "ApplicationSettings".
270                  const XMLCh* xmlch_OptionA
271                        = currentElement->getAttribute(ATTR_OptionA);
272                  m_OptionA = XMLString::transcode(xmlch_OptionA);
273
274                  const XMLCh* xmlch_OptionB
275                        = currentElement->getAttribute(ATTR_OptionB);
276                  m_OptionB = XMLString::transcode(xmlch_OptionB);
277
278                  break;  // Data found. No need to look at other elements in tree.
279               }
280            }
281         }
282      }
283      catch( xercesc::XMLException& e )
284      {
285         char* message = xercesc::XMLString::transcode( e.getMessage() );
286         ostringstream errBuf;
287         errBuf << "Error parsing file: " << message << flush;
288         XMLString::release( &message );
289      }
290   }
291
292   // the following line was added by JMH to get main to run
293   #define MAIN_TEST
294
295   #ifdef MAIN_TEST
296   /* This main is provided for unit test of the class. */
297
298   int main()
299   {
300      string configFile="sample.xml"; // stat file. Get ambigious segfault otherwise.
301
302      GetConfig appConfig;
303
304      appConfig.readConfigFile(configFile);
305
306      cout << "Application option A="  << appConfig.getOptionA()  << endl;
307      cout << "Application option B="  << appConfig.getOptionB()  << endl;
308
309      return 0;
310   }
311   #endif
312
314
315   ========================================================================
```

Setting up the Xerces API for C++ on Mac OS X

Frank Kamayou

91.204 Computing IV, Spring 2012 Semester

March 17, 2012

I decided to follow the source distribution and compile it by myself.

1. The first thing I did was downloading the source code from http://xerces.apache.org/xerces-c/download.cgi

   You can just download the zip file and unzip to a folder (in my case the computing folder).

2. I went to http://xerces.apache.org/xerces-c/install-3.html#Unix and followed the build instructions. I did these three things:

   ./configure CFLAGS="-arch x86_64" CXXFLAGS="-arch x86_64
   ./configure --prefix=/opt
   build: sudo make
   install: sudo make install

   - the sudo commands are important here otherwise it will return some 'permission denied' errors.
   - the --prefix configure option will install development files such as include header files and libraries in "/opt". I got this from (http://www.yolinux.com/TUTORIALS/XML-Xerces-C.html)
   - I also had issues where I got errors because my directory name 'Computing IV' contained a space and I was getting "computing: no such file of directory". So I renamed my directory to just Computing. My guess is that if any directory on the path to xerces-c-3.1.1 is more than just a single word, you might run into issues.

   at the end of installation, I get these warnings:

   make[2]: Nothing to be done for `install-data-am'.
   make[2]: Nothing to be done for `install-exec-am'.

   and after some Googling around, I found out that they are not necessarily a problem, so I kept on going.

The installation ends with:

```
test -z "/opt/lib/pkgconfig" || /opt/local/bin/gmkdir -p "/opt/lib/
pkgconfig"
 /opt/local/bin/ginstall -c -m 644 xerces-c.pc '/opt/lib/pkgconfig'
Francks-MacBook-Pro:xerces-c-3.1.1 franckamayou$
```

so I assume everything went fine.

3. At this point, I picked up at Part II of Jesse's tutorial to build one of the sample programs.

   http://teaching.cs.uml.edu/~heines/91.204/91.204-2011-12s/204-lecs/lecture14.jsp

   EXCEPT that I do not have the g++ issue that breaks the -l option, so no need to worry about his step 6.

4. If you try to compile the NetBeans project now, you will get linker errors such as:

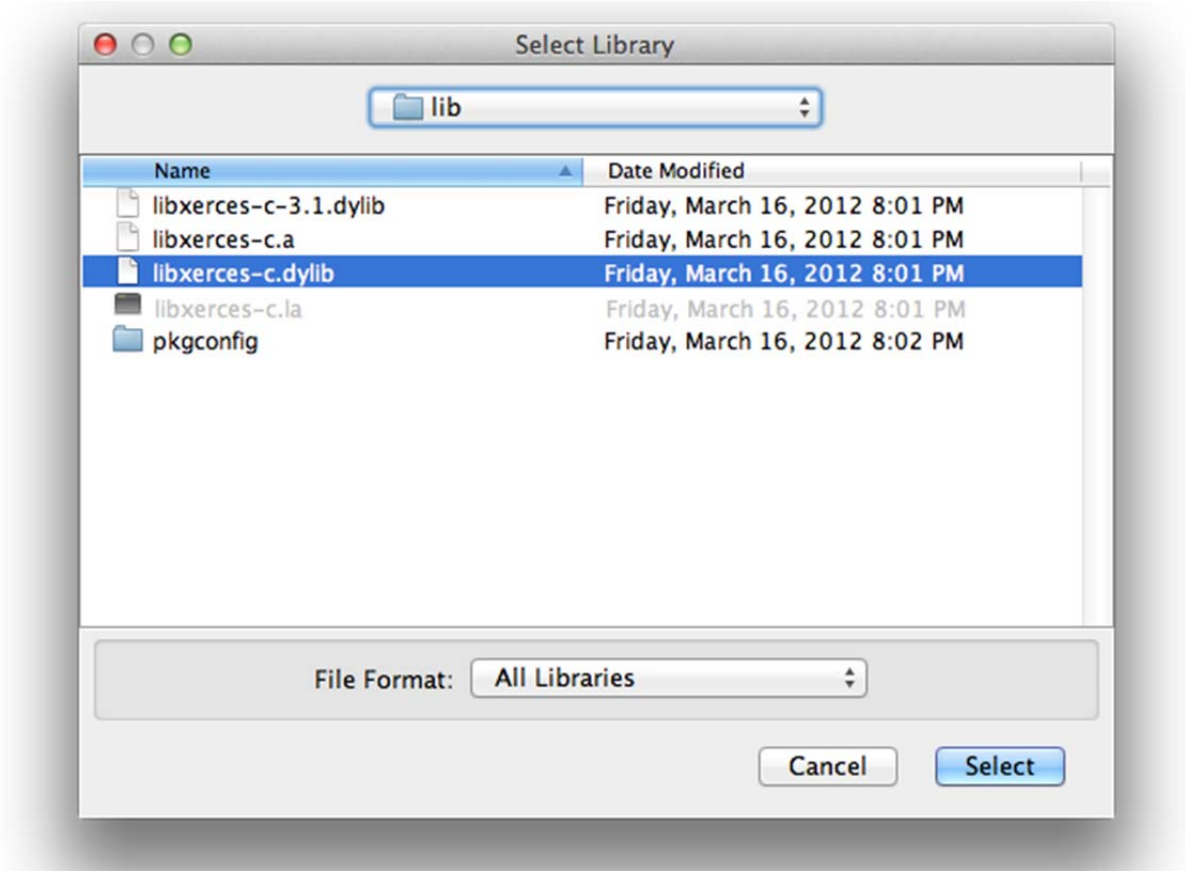
```
g++ -o dist/Debug/GNU-MacOSX/cppapplication_1
build/Debug/GNU-MacOSX/_ext/842610819/DOMPrintErrorHandler.
o build/Debug/GNU-MacOSX/_ext/842610819/DOMPrint.o
build/Debug/GNU-MacOSX/_ext/842610819/DOMTreeErrorReporter
.o build/Debug/GNU-MacOSX/_ext/842610819/DOMPrintFilter.o
Undefined symbols for architecture x86_64:
  "xercesc_3_1::XMLPlatformUtils::fgMemoryManager", referenced
from:
      DOMPrintErrorHandler::handleError(xercesc_3_1::DOMError
const&)    in DOMPrintErrorHandler.o
      _main in DOMPrint.o
      StrX::StrX(unsigned short const*)in DOMPrint.o
      StrX::~StrX() in DOMPrint.o
      StrX::StrX(unsigned short const*)in DOMTreeErrorReporter.o
      StrX::~StrX() in DOMTreeErrorReporter.o
```

   and this is because we need to point NetBeans to the xerces-c library (remember in /opt)

5. Navigate to the project properties (right click on the project) and select the **Build->Linker** category in the left-hand pane. Next, click the **...** button to the right of the **Libraries->Libraries** option in the right-hand pane.

6. In the **Debug->Libraries** dialog box that opens, click the **Add Library...** button. That opens the **Select Library** dialog box (a standard Windows File Open dialog box). You DO have to navigate to the xerces-c library here.

   It's at **/opt/lib/libxerces-c.dylib.**



7. Click **OK** to close that dialog box and you will see **xerces-c** in the Libraries text box in the Project Properties dialog box. Click **OK** again to close that dialog box.

8. The sample program should now compile without error.