```
  1  This file was updated on Tuesday, 2012-02-28 at 11:56 AM
  2
  3
  4  ============================================================================
  5  DOMPrint.cpp
  6  ============================================================================
  7
  8
  9  /*
 10   * Licensed to the Apache Software Foundation (ASF) under one or more
 11   * contributor license agreements.  See the NOTICE file distributed with
 12   * this work for additional information regarding copyright ownership.
 13   * The ASF licenses this file to You under the Apache License, Version 2.0
 14   * (the "License"); you may not use this file except in compliance with
 15   * the License.  You may obtain a copy of the License at
 16   *
 17   *      http://www.apache.org/licenses/LICENSE-2.0
 18   *
 19   * Unless required by applicable law or agreed to in writing, software
 20   * distributed under the License is distributed on an "AS IS" BASIS,
 21   * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 22   * See the License for the specific language governing permissions and
 23   * limitations under the License.
 24   */
 25
 26  /*
 27   * $Id: DOMPrint.cpp 669844 2008-06-20 10:11:44Z borisk $
 28   */
 29
 30  // ---------------------------------------------------------------------------
 31  //  This sample program invokes the XercesDOMParser to build a DOM tree for
 32  //  the specified input file. It then invokes DOMLSSerializer::write() to
 33  //  serialize the resultant DOM tree back to XML stream.
 34  //
 35  //  Note:
 36  //  Application needs to provide its own implementation of
 37  //         DOMErrorHandler (in this sample, the DOMPrintErrorHandler),
 38  //         if it would like to receive notification from the serializer
 39  //         in the case any error occurs during the serialization.
 40  //
 41  //  Application needs to provide its own implementation of
 42  //         DOMLSSerializerFilter (in this sample, the DOMPrintFilter),
 43  //         if it would like to filter out certain part of the DOM
 44  //         representation, but must be aware that thus may render the
 45  //         resultant XML stream invalid.
 46  //
 47  //  Application may choose any combination of characters as the
 48  //         end of line sequence to be used in the resultant XML stream,
 49  //         but must be aware that thus may render the resultant XML
 50  //         stream ill formed.
 51  //
 52  //  Application may choose a particular encoding name in which
 53  //         the output XML stream would be, but must be aware that if
 54  //         characters, unrepresentable in the encoding specified, appearing
 55  //         in markups, may force the serializer to terminate serialization
 56  //         prematurely, and thus no complete serialization would be done.
 57  //
 58  //  Application shall query the serializer first, before set any
 59  //           feature/mode(true, false), or be ready to catch exception if this
 60  //           feature/mode is not supported by the serializer.
 61  //
 62  //  Application needs to clean up the filter, error handler and
 63  //         format target objects created for the serialization.
 64  //
 65  //   Limitations:
 66  //      1.  The encoding="xxx" clause in the XML header should reflect
 67  //          the system local code page, but does not.
```

```
68  //      2.  Cases where the XML data contains characters that can not
69  //          be represented in the system local code page are not handled.
70  //
71  // ---------------------------------------------------------------------------
72
73
74  // ---------------------------------------------------------------------------
75  //  Includes
76  // ---------------------------------------------------------------------------
77  #include <xercesc/util/PlatformUtils.hpp>
78
79  #include <xercesc/dom/DOM.hpp>
80
81  #include <xercesc/framework/StdOutFormatTarget.hpp>
82  #include <xercesc/framework/LocalFileFormatTarget.hpp>
83  #include <xercesc/parsers/XercesDOMParser.hpp>
84  #include <xercesc/util/XMLUni.hpp>
85
86  #include "DOMTreeErrorReporter.hpp"
87  #include "DOMPrintFilter.hpp"
88  #include "DOMPrintErrorHandler.hpp"
89  #include <xercesc/util/OutOfMemoryException.hpp>
90
91  #include <string.h>
92  #include <stdlib.h>
93
94  // ---------------------------------------------------------------------------
95  //  Local data
96  //
97  //  gXmlFile
98  //      The path to the file to parser. Set via command line.
99  //
100 //  gDoNamespaces
101 //      Indicates whether namespace processing should be done.
102 //
103 //  gDoSchema
104 //      Indicates whether schema processing should be done.
105 //
106 //  gSchemaFullChecking
107 //      Indicates whether full schema constraint checking should be done.
108 //
109 //  gDoCreate
110 //      Indicates whether entity reference nodes needs to be created or not
111 //      Defaults to false
112 //
113 //  gOutputEncoding
114 //      The encoding we are to output in. If not set on the command line,
115 //      then it is defaults to the encoding of the input XML file.
116 //
117 //  gSplitCdataSections
118 //      Indicates whether split-cdata-sections is to be enabled or not.
119 //
120 //  gDiscardDefaultContent
121 //      Indicates whether default content is discarded or not.
122 //
123 //  gUseFilter
124 //      Indicates if user wants to plug in the DOMPrintFilter.
125 //
126 //  gValScheme
127 //      Indicates what validation scheme to use. It defaults to 'auto', but
128 //      can be set via the -v= command.
129 //
130 // ---------------------------------------------------------------------------
131 static char*                    gXmlFile              = 0;
132 static bool                     gDoNamespaces         = false;
133 static bool                     gDoSchema             = false;
134 static bool                     gSchemaFullChecking   = false;
```

```
135 static bool                    gDoCreate             = false;
136
137 static char*                   goutputfile           = 0;
138 static char*                   gXPathExpression      = 0;
139
140 // options for DOMLSSerializer's features
141 static XMLCh*                  gOutputEncoding       = 0;
142
143 static bool                    gSplitCdataSections   = true;
144 static bool                    gDiscardDefaultContent = true;
145 static bool                    gUseFilter            = false;
146 static bool                    gFormatPrettyPrint    = false;
147 static bool                    gWriteBOM             = false;
148
149 static XercesDOMParser::ValSchemes    gValScheme      = XercesDOMParser::Val_Auto;
150
151
152 //  Prototypes for internally used functions
153 void usage();
154
155
156 // ---------------------------------------------------------------------------
157 //
158 //  Usage()
159 //
160 // ---------------------------------------------------------------------------
161 void usage()
162 {
163     XERCES_STD_QUALIFIER cout << "\nUsage:\n"
164             "    DOMPrint [options] <XML file>\n\n"
165             "This program invokes the DOM parser, and builds the DOM tree.\n"
166             "It then asks the DOMLSSerializer to serialize the DOM tree.\n"
167             "Options:\n"
168             "    -e          create entity reference nodes. Default is no expansion.\n"
169             "    -v=xxx      Validation scheme [always | never | auto*].\n"
170             "    -n          Enable namespace processing. Default is off.\n"
171             "    -s          Enable schema processing. Default is off.\n"
172             "    -f          Enable full schema constraint checking. Defaults is off.\n"
173             "    -wenc=XXX   Use a particular encoding for output. Default is\n"
174             "                the same encoding as the input XML file. UTF-8 if\n"
175             "                input XML file has not XML declaration.\n"
176             "    -wfile=xxx  Write to a file instead of stdout.\n"
177             "    -wscs=xxx   Enable/Disable split-cdata-sections.      Default on\n"
178             "    -wddc=xxx   Enable/Disable discard-default-content.   Default on\n"
179             "    -wflt=xxx   Enable/Disable filtering.                 Default off\n"
180             "    -wfpp=xxx   Enable/Disable format-pretty-print.       Default off\n"
181             "    -wbom=xxx   Enable/Disable write Byte-Order-Mark      Default off\n"
182             "    -xpath=xxx  Prints only the nodes matching the given XPath.\n"
183             "    -?          Show this help.\n\n"
184             "  * = Default if not provided explicitly.\n\n"
185             "The parser has intrinsic support for the following encodings:\n"
186             "    UTF-8, USASCII, ISO8859-1, UTF-16[BL]E, UCS-4[BL]E,\n"
187             "    WINDOWS-1252, IBM1140, IBM037, IBM1047.\n"
188          <<  XERCES_STD_QUALIFIER endl;
189 }
190
191 // ---------------------------------------------------------------------------
192 //
193 //  main
194 //
195 // ---------------------------------------------------------------------------
196 int main(int argC, char* argV[])
197 {
198     int retval = 0;
199
200     // Initialize the XML4C2 system
201     try
```

```
202     {
203         XMLPlatformUtils::Initialize();
204     }
205
206     catch(const XMLException &toCatch)
207     {
208         XERCES_STD_QUALIFIER cerr << "Error during Xerces-c Initialization.\n"
209              << "  Exception message:"
210              << StrX(toCatch.getMessage()) << XERCES_STD_QUALIFIER endl;
211         return 1;
212     }
213
214     // Check command line and extract arguments.
215     if (argC < 2)
216     {
217         usage();
218         XMLPlatformUtils::Terminate();
219         return 1;
220     }
221
222     // See if non validating dom parser configuration is requested.
223     int parmInd;
224     for (parmInd = 1; parmInd < argC; parmInd++)
225     {
226         // Break out on first parm not starting with a dash
227         if (argV[parmInd][0] != '-')
228             break;
229
230         // Watch for special case help request
231         if (!strcmp(argV[parmInd], "-?"))
232         {
233             usage();
234             XMLPlatformUtils::Terminate();
235             return 2;
236         }
237          else if (!strncmp(argV[parmInd], "-v=", 3)
238               ||  !strncmp(argV[parmInd], "-V=", 3))
239         {
240             const char* const parm = &argV[parmInd][3];
241
242             if (!strcmp(parm, "never"))
243                 gValScheme = XercesDOMParser::Val_Never;
244             else if (!strcmp(parm, "auto"))
245                 gValScheme = XercesDOMParser::Val_Auto;
246             else if (!strcmp(parm, "always"))
247                 gValScheme = XercesDOMParser::Val_Always;
248             else
249             {
250                 XERCES_STD_QUALIFIER cerr << "Unknown -v= value: " << parm << XERCES_STD_QUALIFIER e
251                 XMLPlatformUtils::Terminate();
252                 return 2;
253             }
254         }
255          else if (!strcmp(argV[parmInd], "-n")
256               ||  !strcmp(argV[parmInd], "-N"))
257         {
258             gDoNamespaces = true;
259         }
260          else if (!strcmp(argV[parmInd], "-s")
261               ||  !strcmp(argV[parmInd], "-S"))
262         {
263             gDoSchema = true;
264         }
265          else if (!strcmp(argV[parmInd], "-f")
266               ||  !strcmp(argV[parmInd], "-F"))
267         {
268             gSchemaFullChecking = true;
```

```
269         }
270          else if (!strcmp(argV[parmInd], "-e")
271               ||  !strcmp(argV[parmInd], "-E"))
272         {
273             gDoCreate = true;
274         }
275          else if (!strncmp(argV[parmInd], "-wenc=", 6))
276         {
277              // Get out the encoding name
278              gOutputEncoding = XMLString::transcode( &(argV[parmInd][6]) );
279         }
280          else if (!strncmp(argV[parmInd], "-wfile=", 7))
281         {
282              goutputfile =  &(argV[parmInd][7]);
283         }
284          else if (!strncmp(argV[parmInd], "-wddc=", 6))
285         {
286             const char* const parm = &argV[parmInd][6];
287
288             if (!strcmp(parm, "on"))
289                 gDiscardDefaultContent = true;
290             else if (!strcmp(parm, "off"))
291                 gDiscardDefaultContent = false;
292             else
293             {
294                 XERCES_STD_QUALIFIER cerr << "Unknown -wddc= value: " << parm << XERCES_STD_QUALIFIE
295                 XMLPlatformUtils::Terminate();
296                 return 2;
297             }
298
299         }
300          else if (!strncmp(argV[parmInd], "-wscs=", 6))
301         {
302             const char* const parm = &argV[parmInd][6];
303
304             if (!strcmp(parm, "on"))
305                 gSplitCdataSections = true;
306             else if (!strcmp(parm, "off"))
307                 gSplitCdataSections = false;
308             else
309             {
310                 XERCES_STD_QUALIFIER cerr << "Unknown -wscs= value: " << parm << XERCES_STD_QUALIFIE
311                 XMLPlatformUtils::Terminate();
312                 return 2;
313             }
314         }
315          else if (!strncmp(argV[parmInd], "-wflt=", 6))
316         {
317             const char* const parm = &argV[parmInd][6];
318
319             if (!strcmp(parm, "on"))
320                 gUseFilter = true;
321             else if (!strcmp(parm, "off"))
322                 gUseFilter = false;
323             else
324             {
325                 XERCES_STD_QUALIFIER cerr << "Unknown -wflt= value: " << parm << XERCES_STD_QUALIFIE
326                 XMLPlatformUtils::Terminate();
327                 return 2;
328             }
329         }
330          else if (!strncmp(argV[parmInd], "-wfpp=", 6))
331         {
332             const char* const parm = &argV[parmInd][6];
333
334             if (!strcmp(parm, "on"))
335                 gFormatPrettyPrint = true;
```

```
336             else if (!strcmp(parm, "off"))
337                 gFormatPrettyPrint = false;
338             else
339             {
340                 XERCES_STD_QUALIFIER cerr << "Unknown -wfpp= value: " << parm << XERCES_STD_QUALIFIE
341                 XMLPlatformUtils::Terminate();
342                 return 2;
343             }
344         }
345          else if (!strncmp(argV[parmInd], "-wbom=", 6))
346         {
347             const char* const parm = &argV[parmInd][6];
348
349             if (!strcmp(parm, "on"))
350                 gWriteBOM = true;
351             else if (!strcmp(parm, "off"))
352                 gWriteBOM = false;
353             else
354             {
355                 XERCES_STD_QUALIFIER cerr << "Unknown -wbom= value: " << parm << XERCES_STD_QUALIFIE
356                 XMLPlatformUtils::Terminate();
357                 return 2;
358             }
359         }
360          else if (!strncmp(argV[parmInd], "-xpath=", 7))
361         {
362              gXPathExpression = &(argV[parmInd][7]);
363         }
364          else
365         {
366             XERCES_STD_QUALIFIER cerr << "Unknown option '" << argV[parmInd]
367                  << "', ignoring it.\n" << XERCES_STD_QUALIFIER endl;
368         }
369     }
370
371     //
372     //  And now we have to have only one parameter left and it must be
373     //  the file name.
374     //
375     if (parmInd + 1 != argC)
376     {
377         usage();
378         XMLPlatformUtils::Terminate();
379         return 1;
380     }
381     gXmlFile = argV[parmInd];
382
383     //
384     //  Create our parser, then attach an error handler to the parser.
385     //  The parser will call back to methods of the ErrorHandler if it
386     //  discovers errors during the course of parsing the XML document.
387     //
388     XercesDOMParser *parser = new XercesDOMParser;
389     parser->setValidationScheme(gValScheme);
390     parser->setDoNamespaces(gDoNamespaces);
391     parser->setDoSchema(gDoSchema);
392     parser->setValidationSchemaFullChecking(gSchemaFullChecking);
393     parser->setCreateEntityReferenceNodes(gDoCreate);
394
395     DOMTreeErrorReporter *errReporter = new DOMTreeErrorReporter();
396     parser->setErrorHandler(errReporter);
397
398     //
399     //  Parse the XML file, catching any XML exceptions that might propogate
400     //  out of it.
401     //
402     bool errorsOccured = false;
```

```
403     try
404     {
405         parser->parse(gXmlFile);
406     }
407     catch (const OutOfMemoryException&)
408     {
409         XERCES_STD_QUALIFIER cerr << "OutOfMemoryException" << XERCES_STD_QUALIFIER endl;
410         errorsOccured = true;
411     }
412     catch (const XMLException& e)
413     {
414         XERCES_STD_QUALIFIER cerr << "An error occurred during parsing\n   Message: "
415              << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
416         errorsOccured = true;
417     }
418
419     catch (const DOMException& e)
420     {
421         const unsigned int maxChars = 2047;
422         XMLCh errText[maxChars + 1];
423
424         XERCES_STD_QUALIFIER cerr << "\nDOM Error during parsing: '" << gXmlFile << "'\n"
425              << "DOMException code is:  " << e.code << XERCES_STD_QUALIFIER endl;
426
427         if (DOMImplementation::loadDOMExceptionMsg(e.code, errText, maxChars))
428              XERCES_STD_QUALIFIER cerr << "Message is: " << StrX(errText) << XERCES_STD_QUALIFIER en
429
430         errorsOccured = true;
431     }
432
433     catch (...)
434     {
435         XERCES_STD_QUALIFIER cerr << "An error occurred during parsing\n " << XERCES_STD_QUALIFIER e
436         errorsOccured = true;
437     }
438
439     // If the parse was successful, output the document data from the DOM tree
440     if (!errorsOccured && !errReporter->getSawErrors())
441     {
442         DOMPrintFilter   *myFilter = 0;
443
444         try
445         {
446             // get a serializer, an instance of DOMLSSerializer
447             XMLCh tempStr[3] = {chLatin_L, chLatin_S, chNull};
448             DOMImplementation *impl          = DOMImplementationRegistry::getDOMImplementation(tempS
449             DOMLSSerializer   *theSerializer = ((DOMImplementationLS*)impl)->createLSSerializer();
450             DOMLSOutput       *theOutputDesc = ((DOMImplementationLS*)impl)->createLSOutput();
451
452             // set user specified output encoding
453             theOutputDesc->setEncoding(gOutputEncoding);
454
455             // plug in user's own filter
456             if (gUseFilter)
457             {
458                 // even we say to show attribute, but the DOMLSSerializer
459                 // will not show attribute nodes to the filter as
460                 // the specs explicitly says that DOMLSSerializer shall
461                 // NOT show attributes to DOMLSSerializerFilter.
462                 //
463                 // so DOMNodeFilter::SHOW_ATTRIBUTE has no effect.
464                 // same DOMNodeFilter::SHOW_DOCUMENT_TYPE, no effect.
465                 //
466                 myFilter = new DOMPrintFilter(DOMNodeFilter::SHOW_ELEMENT   |
467                                               DOMNodeFilter::SHOW_ATTRIBUTE |
468                                               DOMNodeFilter::SHOW_DOCUMENT_TYPE);
469                 theSerializer->setFilter(myFilter);
```

```
470             }
471
472             // plug in user's own error handler
473             DOMErrorHandler *myErrorHandler = new DOMPrintErrorHandler();
474             DOMConfiguration* serializerConfig=theSerializer->getDomConfig();
475             serializerConfig->setParameter(XMLUni::fgDOMErrorHandler, myErrorHandler);
476
477             // set feature if the serializer supports the feature/mode
478             if (serializerConfig->canSetParameter(XMLUni::fgDOMWRTSplitCdataSections, gSplitCdataSec
479                 serializerConfig->setParameter(XMLUni::fgDOMWRTSplitCdataSections, gSplitCdataSectio
480
481             if (serializerConfig->canSetParameter(XMLUni::fgDOMWRTDiscardDefaultContent, gDiscardDef
482                 serializerConfig->setParameter(XMLUni::fgDOMWRTDiscardDefaultContent, gDiscardDefaul
483
484             if (serializerConfig->canSetParameter(XMLUni::fgDOMWRTFormatPrettyPrint, gFormatPrettyPr
485                 serializerConfig->setParameter(XMLUni::fgDOMWRTFormatPrettyPrint, gFormatPrettyPrint
486
487             if (serializerConfig->canSetParameter(XMLUni::fgDOMWRTBOM, gWriteBOM))
488                 serializerConfig->setParameter(XMLUni::fgDOMWRTBOM, gWriteBOM);
489
490             //
491             // Plug in a format target to receive the resultant
492             // XML stream from the serializer.
493             //
494             // StdOutFormatTarget prints the resultant XML stream
495             // to stdout once it receives any thing from the serializer.
496             //
497             XMLFormatTarget *myFormTarget;
498             if (goutputfile)
499                 myFormTarget=new LocalFileFormatTarget(goutputfile);
500             else
501                 myFormTarget=new StdOutFormatTarget();
502             theOutputDesc->setByteStream(myFormTarget);
503
504             // get the DOM representation
505             DOMDocument *doc = parser->getDocument();
506
507             //
508             // do the serialization through DOMLSSerializer::write();
509             //
510             if(gXPathExpression!=NULL)
511             {
512                 XMLCh* xpathStr=XMLString::transcode(gXPathExpression);
513                 DOMElement* root = doc->getDocumentElement();
514                 try
515                 {
516                     DOMXPathNSResolver* resolver=doc->createNSResolver(root);
517                     DOMXPathResult* result=doc->evaluate(
518                       xpathStr,
519                       root,
520                       resolver,
521                       DOMXPathResult::ORDERED_NODE_SNAPSHOT_TYPE,
522                       NULL);
523
524                     XMLSize_t nLength = result->getSnapshotLength();
525                     for(XMLSize_t i = 0; i < nLength; i++)
526                     {
527                       result->snapshotItem(i);
528                       theSerializer->write(result->getNodeValue(), theOutputDesc);
529                     }
530
531                     result->release();
532                     resolver->release ();
533                 }
534                 catch(const DOMXPathException& e)
535                 {
536                     XERCES_STD_QUALIFIER cerr << "An error occurred during processing of the XPath e
```

```
537                         << XERCES_STD_QUALIFIER endl
538                         << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
539                     retval = 4;
540                 }
541                 catch(const DOMException& e)
542                 {
543                     XERCES_STD_QUALIFIER cerr << "An error occurred during processing of the XPath e
544                         << XERCES_STD_QUALIFIER endl
545                         << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
546                     retval = 4;
547                 }
548                 XMLString::release(&xpathStr);
549             }
550             else
551                 theSerializer->write(doc, theOutputDesc);
552
553             theOutputDesc->release();
554             theSerializer->release();
555
556             //
557             // Filter, formatTarget and error handler
558             // are NOT owned by the serializer.
559             //
560             delete myFormTarget;
561             delete myErrorHandler;
562
563             if (gUseFilter)
564                 delete myFilter;
565
566         }
567         catch (const OutOfMemoryException&)
568         {
569             XERCES_STD_QUALIFIER cerr << "OutOfMemoryException" << XERCES_STD_QUALIFIER endl;
570             retval = 5;
571         }
572         catch (XMLException& e)
573         {
574             XERCES_STD_QUALIFIER cerr << "An error occurred during creation of output transcoder. Ms
575                 << XERCES_STD_QUALIFIER endl
576                 << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
577             retval = 4;
578         }
579
580     }
581     else
582         retval = 4;
583
584     //
585     //  Clean up the error handler. The parser does not adopt handlers
586     //  since they could be many objects or one object installed for multiple
587     //  handlers.
588     //
589     delete errReporter;
590
591     //
592     //  Delete the parser itself.  Must be done prior to calling Terminate, below.
593     //
594     delete parser;
595
596     XMLString::release(&gOutputEncoding);
597
598     // And call the termination method
599     XMLPlatformUtils::Terminate();
600
601     return retval;
602 }
603
```

```
604
605 ============================================================================
606 DOMPrintErrorHandler.hpp
607 ============================================================================
608
609
610 /*
611  * Licensed to the Apache Software Foundation (ASF) under one or more
612  * contributor license agreements.  See the NOTICE file distributed with
613  * this work for additional information regarding copyright ownership.
614  * The ASF licenses this file to You under the Apache License, Version 2.0
615  * (the "License"); you may not use this file except in compliance with
616  * the License.  You may obtain a copy of the License at
617  *
618  *      http://www.apache.org/licenses/LICENSE-2.0
619  *
620  * Unless required by applicable law or agreed to in writing, software
621  * distributed under the License is distributed on an "AS IS" BASIS,
622  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
623  * See the License for the specific language governing permissions and
624  * limitations under the License.
625  */
626
627 /*
628  * $Id: DOMPrintErrorHandler.hpp 471735 2006-11-06 13:53:58Z amassari $
629  */
630
631
632 #ifndef DOM_PRINT_ERROR_HANDLER_HPP
633 #define DOM_PRINT_ERROR_HANDLER_HPP
634
635 #include <xercesc/dom/DOMErrorHandler.hpp>
636
637 XERCES_CPP_NAMESPACE_USE
638
639 class DOMPrintErrorHandler : public DOMErrorHandler
640 {
641 public:
642
643     DOMPrintErrorHandler(){};
644     ~DOMPrintErrorHandler(){};
645
646     /** @name The error handler interface */
647     bool handleError(const DOMError& domError);
648     void resetErrors(){};
649
650 private :
651     /* Unimplemented constructors and operators */
652     DOMPrintErrorHandler(const DOMErrorHandler&);
653     void operator=(const DOMErrorHandler&);
654
655 };
656
657 #endif
658
659
```

```
660 ============================================================================
661 DOMPrintErrorHandler.cpp
662 ============================================================================
663
664
665 /*
666  * Licensed to the Apache Software Foundation (ASF) under one or more
667  * contributor license agreements.  See the NOTICE file distributed with
668  * this work for additional information regarding copyright ownership.
669  * The ASF licenses this file to You under the Apache License, Version 2.0
670  * (the "License"); you may not use this file except in compliance with
671  * the License.  You may obtain a copy of the License at
672  *
673  *      http://www.apache.org/licenses/LICENSE-2.0
674  *
675  * Unless required by applicable law or agreed to in writing, software
676  * distributed under the License is distributed on an "AS IS" BASIS,
677  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
678  * See the License for the specific language governing permissions and
679  * limitations under the License.
680  */
681
682 /*
683  * $Id: DOMPrintErrorHandler.cpp 471735 2006-11-06 13:53:58Z amassari $
684  */
685
686 #include <xercesc/util/XMLString.hpp>
687 #include <xercesc/dom/DOMError.hpp>
688 #if defined(XERCES_NEW_IOSTREAMS)
689 #include <iostream>
690 #else
691 #include <iostream.h>
692 #endif
693
694 #include "DOMPrintErrorHandler.hpp"
695
696 bool DOMPrintErrorHandler::handleError(const DOMError &domError)
697 {
698     // Display whatever error message passed from the serializer
699     if (domError.getSeverity() == DOMError::DOM_SEVERITY_WARNING)
700         XERCES_STD_QUALIFIER cerr << "\nWarning Message: ";
701     else if (domError.getSeverity() == DOMError::DOM_SEVERITY_ERROR)
702         XERCES_STD_QUALIFIER cerr << "\nError Message: ";
703     else
704         XERCES_STD_QUALIFIER cerr << "\nFatal Message: ";
705
706     char *msg = XMLString::transcode(domError.getMessage());
707     XERCES_STD_QUALIFIER cerr<< msg <<XERCES_STD_QUALIFIER endl;
708     XMLString::release(&msg);
709
710     // Instructs the serializer to continue serialization if possible.
711     return true;
712 }
713
714
715
716 ============================================================================
717 DOMPrintFilter.hpp
718 ============================================================================
719
720
721 /*
722  * Licensed to the Apache Software Foundation (ASF) under one or more
723  * contributor license agreements.  See the NOTICE file distributed with
724  * this work for additional information regarding copyright ownership.
725  * The ASF licenses this file to You under the Apache License, Version 2.0
726  * (the "License"); you may not use this file except in compliance with
```

```
 * the License.  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/*
 * $Id: DOMPrintFilter.hpp 671894 2008-06-26 13:29:21Z borisk $
 */

////////////////////////////////////////////////////////////////////
// DOMPrintFilter.hpp: a sample implementation of DOMWriterFilter.
//
////////////////////////////////////////////////////////////////////

#ifndef DOMPrintFilter_HEADER_GUARD_
#define DOMPrintFilter_HEADER_GUARD_

#include <xercesc/dom/DOMLSSerializerFilter.hpp>

XERCES_CPP_NAMESPACE_USE

class DOMPrintFilter : public DOMLSSerializerFilter {
public:

    DOMPrintFilter(ShowType whatToShow = DOMNodeFilter::SHOW_ALL);
    ~DOMPrintFilter(){};

    virtual FilterAction acceptNode(const DOMNode*) const;
    virtual ShowType getWhatToShow() const {return fWhatToShow;};

private:
    // unimplemented copy ctor and assignement operator
    DOMPrintFilter(const DOMPrintFilter&);
    DOMPrintFilter & operator = (const DOMPrintFilter&);

    ShowType fWhatToShow;
};

#endif


=====================================================================
DOMPrintFilter.cpp
=====================================================================


/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
```

```
794   */
795
796  /*
797   * $Id: DOMPrintFilter.cpp 671894 2008-06-26 13:29:21Z borisk $
798   */
799
800  #include "DOMPrintFilter.hpp"
801  #include <xercesc/util/XMLUniDefs.hpp>
802  #include <xercesc/util/XMLString.hpp>
803
804  static const XMLCh  element_person[]=
805  {
806      chLatin_p, chLatin_e, chLatin_r, chLatin_s, chLatin_o, chLatin_n, chNull
807  };
808
809  static const XMLCh  element_link[]=
810  {
811      chLatin_l, chLatin_i, chLatin_n, chLatin_k, chNull
812  };
813
814  DOMPrintFilter::DOMPrintFilter(ShowType whatToShow)
815  :fWhatToShow(whatToShow)
816  {}
817
818  DOMNodeFilter::FilterAction DOMPrintFilter::
819  acceptNode(const DOMNode* node) const
820  {
821      //
822      // The DOMLSSerializer shall call getWhatToShow() before calling
823      // acceptNode(), to show nodes which are supposed to be
824      // shown to this filter.
825      //
826      // REVISIT: In case the DOMLSSerializer does not follow the protocol,
827      //          Shall the filter honour, or NOT, what it claims
828      //          it is interested in ?
829      //
830      // The DOMLS specs does not specify that acceptNode() shall do
831      // this way, or not, so it is up the implementation,
832      // to skip the code below for the sake of performance ...
833      //
834      if ((getWhatToShow() & (1 << (node->getNodeType() - 1))) == 0)
835          return DOMNodeFilter::FILTER_ACCEPT;
836
837      switch (node->getNodeType())
838      {
839      case DOMNode::ELEMENT_NODE:
840          {
841              // for element whose name is "person", skip it
842              if (XMLString::compareString(node->getNodeName(), element_person)==0)
843                  return DOMNodeFilter::FILTER_SKIP;
844              // for element whose name is "line", reject it
845              if (XMLString::compareString(node->getNodeName(), element_link)==0)
846                  return DOMNodeFilter::FILTER_REJECT;
847              // for rest, accept it
848              return DOMNodeFilter::FILTER_ACCEPT;
849
850              break;
851          }
852      case DOMNode::COMMENT_NODE:
853          {
854              // the WhatToShow will make this no effect
855              return DOMNodeFilter::FILTER_REJECT;
856              break;
857          }
858      case DOMNode::TEXT_NODE:
859          {
860              // the WhatToShow will make this no effect
```

```
861             return DOMNodeFilter::FILTER_REJECT;
862             break;
863         }
864     case DOMNode::DOCUMENT_TYPE_NODE:
865         {
866             // even we say we are going to process document type,
867             // we are not able be to see this node since
868             // DOMLSSerializerImpl (a XercesC's default implementation
869             // of DOMLSSerializer) will not pass DocumentType node to
870             // this filter.
871             //
872             return DOMNodeFilter::FILTER_REJECT;  // no effect
873             break;
874         }
875     case DOMNode::DOCUMENT_NODE:
876         {
877             // same as DOCUMENT_NODE
878             return DOMNodeFilter::FILTER_REJECT;  // no effect
879             break;
880         }
881     default :
882         {
883             return DOMNodeFilter::FILTER_ACCEPT;
884             break;
885         }
886     }
887
888     return DOMNodeFilter::FILTER_ACCEPT;
889 }
890
891
892 //============================================================================
893 // DOMTreeErrorReporter.hpp
894 //============================================================================
895
896
897 /*
898  * Licensed to the Apache Software Foundation (ASF) under one or more
899  * contributor license agreements.  See the NOTICE file distributed with
900  * this work for additional information regarding copyright ownership.
901  * The ASF licenses this file to You under the Apache License, Version 2.0
902  * (the "License"); you may not use this file except in compliance with
903  * the License.  You may obtain a copy of the License at
904  *
905  *      http://www.apache.org/licenses/LICENSE-2.0
906  *
907  * Unless required by applicable law or agreed to in writing, software
908  * distributed under the License is distributed on an "AS IS" BASIS,
909  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
910  * See the License for the specific language governing permissions and
911  * limitations under the License.
912  */
913
914 /*
915  * $Id: DOMTreeErrorReporter.hpp 471735 2006-11-06 13:53:58Z amassari $
916  */
917
918 #include <xercesc/util/XercesDefs.hpp>
919 #include <xercesc/sax/ErrorHandler.hpp>
920 #if defined(XERCES_NEW_IOSTREAMS)
921 #include <iostream>
922 #else
923 #include <iostream.h>
924 #endif
925
926
927 XERCES_CPP_NAMESPACE_USE
```

```
928
929
930  class DOMTreeErrorReporter : public ErrorHandler
931  {
932  public:
933      // -----------------------------------------------------------------------
934      //  Constructors and Destructor
935      // -----------------------------------------------------------------------
936      DOMTreeErrorReporter() :
937         fSawErrors(false)
938      {
939      }
940
941      ~DOMTreeErrorReporter()
942      {
943      }
944
945
946      // -----------------------------------------------------------------------
947      //  Implementation of the error handler interface
948      // -----------------------------------------------------------------------
949      void warning(const SAXParseException& toCatch);
950      void error(const SAXParseException& toCatch);
951      void fatalError(const SAXParseException& toCatch);
952      void resetErrors();
953
954      // -----------------------------------------------------------------------
955      //  Getter methods
956      // -----------------------------------------------------------------------
957      bool getSawErrors() const;
958
959      // -----------------------------------------------------------------------
960      //  Private data members
961      //
962      //  fSawErrors
963      //      This is set if we get any errors, and is queryable via a getter
964      //      method. Its used by the main code to suppress output if there are
965      //      errors.
966      // -----------------------------------------------------------------------
967      bool    fSawErrors;
968  };
969
970  inline bool DOMTreeErrorReporter::getSawErrors() const
971  {
972      return fSawErrors;
973  }
974
975  // ---------------------------------------------------------------------------
976  //  This is a simple class that lets us do easy (though not terribly efficient)
977  //  trancoding of XMLCh data to local code page for display.
978  // ---------------------------------------------------------------------------
979  class StrX
980  {
981  public :
982      // -----------------------------------------------------------------------
983      //  Constructors and Destructor
984      // -----------------------------------------------------------------------
985      StrX(const XMLCh* const toTranscode)
986      {
987          // Call the private transcoding method
988          fLocalForm = XMLString::transcode(toTranscode);
989      }
990
991      ~StrX()
992      {
993          XMLString::release(&fLocalForm);
994      }
```

```
 995
 996
 997     // ---------------------------------------------------------------------
 998     //  Getter methods
 999     // ---------------------------------------------------------------------
1000     const char* localForm() const
1001     {
1002         return fLocalForm;
1003     }
1004
1005 private :
1006     // ---------------------------------------------------------------------
1007     //  Private data members
1008     //
1009     //  fLocalForm
1010     //      This is the local code page form of the string.
1011     // ---------------------------------------------------------------------
1012     char*   fLocalForm;
1013 };
1014
1015 inline XERCES_STD_QUALIFIER ostream& operator<<(XERCES_STD_QUALIFIER ostream& target, const StrX& to
1016 {
1017     target << toDump.localForm();
1018     return target;
1019 }
1020
1021
1022
1023 ============================================================================
1024 DOMTreeErrorReporter.cpp
1025 ============================================================================
1026
1027
1028 /*
1029  * Licensed to the Apache Software Foundation (ASF) under one or more
1030  * contributor license agreements.  See the NOTICE file distributed with
1031  * this work for additional information regarding copyright ownership.
1032  * The ASF licenses this file to You under the Apache License, Version 2.0
1033  * (the "License"); you may not use this file except in compliance with
1034  * the License.  You may obtain a copy of the License at
1035  *
1036  *      http://www.apache.org/licenses/LICENSE-2.0
1037  *
1038  * Unless required by applicable law or agreed to in writing, software
1039  * distributed under the License is distributed on an "AS IS" BASIS,
1040  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
1041  * See the License for the specific language governing permissions and
1042  * limitations under the License.
1043  */
1044
1045 /*
1046  * $Id: DOMTreeErrorReporter.cpp 471735 2006-11-06 13:53:58Z amassari $
1047  */
1048
1049 // ---------------------------------------------------------------------------
1050 //  Includes
1051 // ---------------------------------------------------------------------------
1052 #include <xercesc/sax/SAXParseException.hpp>
1053 #include "DOMTreeErrorReporter.hpp"
1054 #if defined(XERCES_NEW_IOSTREAMS)
1055 #include <iostream>
1056 #else
1057 #include <iostream.h>
1058 #endif
1059 #include <stdlib.h>
1060 #include <string.h>
1061
```

```
1062
1063  void DOMTreeErrorReporter::warning(const SAXParseException&)
1064  {
1065      //
1066      // Ignore all warnings.
1067      //
1068  }
1069
1070  void DOMTreeErrorReporter::error(const SAXParseException& toCatch)
1071  {
1072      fSawErrors = true;
1073      XERCES_STD_QUALIFIER cerr << "Error at file \"" << StrX(toCatch.getSystemId())
1074           << "\", line " << toCatch.getLineNumber()
1075           << ", column " << toCatch.getColumnNumber()
1076           << "\n   Message: " << StrX(toCatch.getMessage()) << XERCES_STD_QUALIFIER endl;
1077  }
1078
1079  void DOMTreeErrorReporter::fatalError(const SAXParseException& toCatch)
1080  {
1081      fSawErrors = true;
1082      XERCES_STD_QUALIFIER cerr << "Fatal Error at file \"" << StrX(toCatch.getSystemId())
1083           << "\", line " << toCatch.getLineNumber()
1084           << ", column " << toCatch.getColumnNumber()
1085           << "\n   Message: " << StrX(toCatch.getMessage()) << XERCES_STD_QUALIFIER endl;
1086  }
1087
1088  void DOMTreeErrorReporter::resetErrors()
1089  {
1090      fSawErrors = false;
1091  }
1092
1093
1094
1095  ============================================================================
```