```
This file was updated on Monday, 2014-11-03 at 2:51 PM


====================================================================
SAX2Count.hpp
====================================================================


/*
 * 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.
 */

/*
 * $Id: SAX2Count.hpp 471735 2006-11-06 13:53:58Z amassari $
 */


// ---------------------------------------------------------------------------
//  Includes for all the program files to see
// ---------------------------------------------------------------------------

#include <xercesc/util/PlatformUtils.hpp>
#include <stdlib.h>
#include <string.h>
#if defined(XERCES_NEW_IOSTREAMS)
#include <iostream>
#else
#include <iostream.h>
#endif
#include "SAX2CountHandlers.hpp"
#include <xercesc/sax2/XMLReaderFactory.hpp>
#include <xercesc/sax2/SAX2XMLReader.hpp>


// ---------------------------------------------------------------------------
//  This is a simple class that lets us do easy (though not terribly efficient)
//  trancoding of XMLCh data to local code page for display.
// ---------------------------------------------------------------------------
class StrX
{
public :
    // -----------------------------------------------------------------------
    //  Constructors and Destructor
    // -----------------------------------------------------------------------
    StrX(const XMLCh* const toTranscode)
    {
        // Call the private transcoding method
        fLocalForm = XMLString::transcode(toTranscode);
    }

    ~StrX()
    {
        XMLString::release(&fLocalForm);
    }

```

```
69      // ---------------------------------------------------------------------
70      //  Getter methods
71      // ---------------------------------------------------------------------
72      const char* localForm() const
73      {
74          return fLocalForm;
75      }
76
77  private :
78      // ---------------------------------------------------------------------
79      //  Private data members
80      //
81      //  fLocalForm
82      //      This is the local code page form of the string.
83      // ---------------------------------------------------------------------
84      char*   fLocalForm;
85  };
86
87  inline XERCES_STD_QUALIFIER ostream& operator<<(XERCES_STD_QUALIFIER ostream& target, const StrX& toDump)
88  {
89      target << toDump.localForm();
90      return target;
91  }
92
93
94  ============================================================================
95  SAX2Count.cpp
96  ============================================================================
97
98
99  /*
100  * Licensed to the Apache Software Foundation (ASF) under one or more
101  * contributor license agreements.  See the NOTICE file distributed with
102  * this work for additional information regarding copyright ownership.
103  * The ASF licenses this file to You under the Apache License, Version 2.0
104  * (the "License"); you may not use this file except in compliance with
105  * the License.  You may obtain a copy of the License at
106  *
107  *      http://www.apache.org/licenses/LICENSE-2.0
108  *
109  * Unless required by applicable law or agreed to in writing, software
110  * distributed under the License is distributed on an "AS IS" BASIS,
111  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
112  * See the License for the specific language governing permissions and
113  * limitations under the License.
114  */
115
116 /*
117  * $Id: SAX2Count.cpp 833057 2009-11-05 15:25:10Z borisk $
118  */
119
120
121 // ---------------------------------------------------------------------------
122 //  Includes
123 // ---------------------------------------------------------------------------
124 #include "SAX2Count.hpp"
125 #include <xercesc/util/PlatformUtils.hpp>
126 #include <xercesc/sax2/SAX2XMLReader.hpp>
127 #include <xercesc/sax2/XMLReaderFactory.hpp>
128 #if defined(XERCES_NEW_IOSTREAMS)
129 #include <fstream>
130 #else
131 #include <fstream.h>
132 #endif
133 #include <xercesc/util/OutOfMemoryException.hpp>
134
```

```
135 // ---------------------------------------------------------------------------
136 //  Local helper methods
137 // ---------------------------------------------------------------------------
138 void usage()
139 {
140     XERCES_STD_QUALIFIER cout << "\nUsage:\n"
141             "    SAX2Count [options] <XML file | List file>\n\n"
142             "This program invokes the SAX2XMLReader, and then prints the\n"
143             "number of elements, attributes, spaces and characters found\n"
144             "in each XML file, using SAX2 API.\n\n"
145             "Options:\n"
146             "    -l          Indicate the input file is a List File that has a list of xml files.\n"
147             "                Default to off (Input file is an XML file).\n"
148             "    -v=xxx      Validation scheme [always | never | auto*].\n"
149             "    -f          Enable full schema constraint checking processing. Defaults to off.\n"
150             "    -p          Enable namespace-prefixes feature. Defaults to off.\n"
151             "    -n          Disable namespace processing. Defaults to on.\n"
152             "                NOTE: THIS IS OPPOSITE FROM OTHER SAMPLES.\n"
153             "    -s          Disable schema processing. Defaults to on.\n"
154             "                NOTE: THIS IS OPPOSITE FROM OTHER SAMPLES.\n"
155             "    -i          Disable identity constraint checking. Defaults to on.\n"
156             "                NOTE: THIS IS OPPOSITE FROM OTHER SAMPLES.\n"
157             "    -locale=ll_CC specify the locale, default: en_US.\n"
158             "    -?          Show this help.\n\n"
159             "  * = Default if not provided explicitly.\n"
160          << XERCES_STD_QUALIFIER endl;
161 }
162
163
164 // ---------------------------------------------------------------------------
165 //  Program entry point
166 // ---------------------------------------------------------------------------
167 int main(int argC, char* argV[])
168 {
169
170     // Check command line and extract arguments.
171     if (argC < 2)
172     {
173         usage();
174         return 1;
175     }
176
177     const char*                 xmlFile      = 0;
178     SAX2XMLReader::ValSchemes   valScheme    = SAX2XMLReader::Val_Auto;
179     bool                        doNamespaces = true;
180     bool                        doSchema = true;
181     bool                        schemaFullChecking = false;
182     bool                        identityConstraintChecking = true;
183     bool                        doList = false;
184     bool                        errorOccurred = false;
185     bool                        namespacePrefixes = false;
186     bool                        recognizeNEL = false;
187     char                        localeStr[64];
188     memset(localeStr, 0, sizeof localeStr);
189
190     int argInd;
191     for (argInd = 1; argInd < argC; argInd++)
192     {
193         // Break out on first parm not starting with a dash
194         if (argV[argInd][0] != '-')
195             break;
196
197         // Watch for special case help request
198         if (!strcmp(argV[argInd], "-?"))
199         {
200             usage();
201             return 2;
202         }
```

```
203           else if (!strncmp(argV[argInd], "-v=", 3)
204                ||  !strncmp(argV[argInd], "-V=", 3))
205           {
206               const char* const parm = &argV[argInd][3];
207
208               if (!strcmp(parm, "never"))
209                   valScheme = SAX2XMLReader::Val_Never;
210               else if (!strcmp(parm, "auto"))
211                   valScheme = SAX2XMLReader::Val_Auto;
212               else if (!strcmp(parm, "always"))
213                   valScheme = SAX2XMLReader::Val_Always;
214               else
215               {
216                   XERCES_STD_QUALIFIER cerr << "Unknown -v= value: " << parm << XERCES_STD_QUALIFIER endl;
217                   return 2;
218               }
219           }
220           else if (!strcmp(argV[argInd], "-n")
221                ||  !strcmp(argV[argInd], "-N"))
222           {
223               doNamespaces = false;
224           }
225           else if (!strcmp(argV[argInd], "-s")
226                ||  !strcmp(argV[argInd], "-S"))
227           {
228               doSchema = false;
229           }
230           else if (!strcmp(argV[argInd], "-f")
231                ||  !strcmp(argV[argInd], "-F"))
232           {
233               schemaFullChecking = true;
234           }
235           else if (!strcmp(argV[argInd], "-i")
236                ||  !strcmp(argV[argInd], "-I"))
237           {
238               identityConstraintChecking = false;
239           }
240           else if (!strcmp(argV[argInd], "-l")
241                ||  !strcmp(argV[argInd], "-L"))
242           {
243               doList = true;
244           }
245           else if (!strcmp(argV[argInd], "-p")
246                ||  !strcmp(argV[argInd], "-P"))
247           {
248               namespacePrefixes = true;
249           }
250           else if (!strcmp(argV[argInd], "-special:nel"))
251           {
252               // turning this on will lead to non-standard compliance behaviour
253               // it will recognize the unicode character 0x85 as new line character
254               // instead of regular character as specified in XML 1.0
255               // do not turn this on unless really necessary
256                recognizeNEL = true;
257           }
258           else if (!strncmp(argV[argInd], "-locale=", 8))
259           {
260                // Get out the end of line
261                strncpy(localeStr, &(argV[argInd][8]), sizeof localeStr);
262           }
263           else
264           {
265               XERCES_STD_QUALIFIER cerr << "Unknown option '" << argV[argInd]
266                   << "', ignoring it\n" << XERCES_STD_QUALIFIER endl;
267           }
268       }
269
```

```
270     //
271     //  There should be only one and only one parameter left, and that
272     //  should be the file name.
273     //
274     if (argInd != argC - 1)
275     {
276         usage();
277         return 1;
278     }
279
280     // Initialize the XML4C2 system
281     try
282     {
283         if (strlen(localeStr))
284         {
285             XMLPlatformUtils::Initialize(localeStr);
286         }
287         else
288         {
289             XMLPlatformUtils::Initialize();
290         }
291
292         if (recognizeNEL)
293         {
294             XMLPlatformUtils::recognizeNEL(recognizeNEL);
295         }
296     }
297
298     catch (const XMLException& toCatch)
299     {
300         XERCES_STD_QUALIFIER cerr << "Error during initialization! Message:\n"
301             << StrX(toCatch.getMessage()) << XERCES_STD_QUALIFIER endl;
302         return 1;
303     }
304
305     //
306     //  Create a SAX parser object. Then, according to what we were told on
307     //  the command line, set it to validate or not.
308     //
309     SAX2XMLReader* parser = XMLReaderFactory::createXMLReader();
310     parser->setFeature(XMLUni::fgSAX2CoreNameSpaces, doNamespaces);
311     parser->setFeature(XMLUni::fgXercesSchema, doSchema);
312     parser->setFeature(XMLUni::fgXercesHandleMultipleImports, true);
313     parser->setFeature(XMLUni::fgXercesSchemaFullChecking, schemaFullChecking);
314     parser->setFeature(XMLUni::fgXercesIdentityConstraintChecking, identityConstraintChecking);
315     parser->setFeature(XMLUni::fgSAX2CoreNameSpacePrefixes, namespacePrefixes);
316
317     if (valScheme == SAX2XMLReader::Val_Auto)
318     {
319         parser->setFeature(XMLUni::fgSAX2CoreValidation, true);
320         parser->setFeature(XMLUni::fgXercesDynamic, true);
321     }
322     if (valScheme == SAX2XMLReader::Val_Never)
323     {
324         parser->setFeature(XMLUni::fgSAX2CoreValidation, false);
325     }
326     if (valScheme == SAX2XMLReader::Val_Always)
327     {
328         parser->setFeature(XMLUni::fgSAX2CoreValidation, true);
329         parser->setFeature(XMLUni::fgXercesDynamic, false);
330     }
331
332     //
333     //  Create our SAX handler object and install it on the parser, as the
334     //  document and error handler.
335     //
336     SAX2CountHandlers handler;
337     parser->setContentHandler(&handler);
338     parser->setErrorHandler(&handler);
339
```

```
340     //
341     //  Get the starting time and kick off the parse of the indicated
342     //  file. Catch any exceptions that might propogate out of it.
343     //
344     unsigned long duration;
345
346     bool more = true;
347     XERCES_STD_QUALIFIER ifstream fin;
348
349     // the input is a list file
350     if (doList)
351         fin.open(argV[argInd]);
352
353     if (fin.fail()) {
354         XERCES_STD_QUALIFIER cerr <<"Cannot open the list file: " << argV[argInd] << XERCES_STD_QUALIFIER en
355         return 2;
356     }
357
358     while (more)
359     {
360         char fURI[1000];
361         //initialize the array to zeros
362         memset(fURI,0,sizeof(fURI));
363
364         if (doList) {
365             if (! fin.eof() ) {
366                 fin.getline (fURI, sizeof(fURI));
367                 if (!*fURI)
368                     continue;
369                 else {
370                     xmlFile = fURI;
371                     XERCES_STD_QUALIFIER cerr << "==Parsing== " << xmlFile << XERCES_STD_QUALIFIER endl;
372                 }
373             }
374             else
375                 break;
376         }
377         else {
378             xmlFile = argV[argInd];
379             more = false;
380         }
381
382         //reset error count first
383         handler.resetErrors();
384
385         try
386         {
387             const unsigned long startMillis = XMLPlatformUtils::getCurrentMillis();
388             parser->parse(xmlFile);
389             const unsigned long endMillis = XMLPlatformUtils::getCurrentMillis();
390             duration = endMillis - startMillis;
391         }
392         catch (const OutOfMemoryException&)
393         {
394             XERCES_STD_QUALIFIER cerr << "OutOfMemoryException" << XERCES_STD_QUALIFIER endl;
395             errorOccurred = true;
396             continue;
397         }
398         catch (const XMLException& e)
399         {
400             XERCES_STD_QUALIFIER cerr << "\nError during parsing: '" << xmlFile << "'\n"
401                 << "Exception message is:  \n"
402                 << StrX(e.getMessage()) << "\n" << XERCES_STD_QUALIFIER endl;
403             errorOccurred = true;
404             continue;
405         }
406
```

```
407         catch (...)
408         {
409             XERCES_STD_QUALIFIER cerr << "\nUnexpected exception during parsing: '" << xmlFile << "'\n";
410             errorOccurred = true;
411             continue;
412         }
413 
414 
415         // Print out the stats that we collected and time taken
416         if (!handler.getSawErrors())
417         {
418             XERCES_STD_QUALIFIER cout << xmlFile << ": " << duration << " ms ("
419                 << handler.getElementCount() << " elems, "
420                 << handler.getAttrCount() << " attrs, "
421                 << handler.getSpaceCount() << " spaces, "
422                 << handler.getCharacterCount() << " chars)" << XERCES_STD_QUALIFIER endl;
423         }
424         else
425             errorOccurred = true;
426     }
427 
428     if (doList)
429         fin.close();
430 
431     //
432     //  Delete the parser itself.  Must be done prior to calling Terminate, below.
433     //
434     delete parser;
435 
436     // And call the termination method
437     XMLPlatformUtils::Terminate();
438 
439     if (errorOccurred)
440         return 4;
441     else
442         return 0;
443 
444 }
445 
446 
447 ===========================================================================
448 SAX2CountHandlers.hpp
449 ===========================================================================
450 
451 
452 /*
453  * Licensed to the Apache Software Foundation (ASF) under one or more
454  * contributor license agreements.  See the NOTICE file distributed with
455  * this work for additional information regarding copyright ownership.
456  * The ASF licenses this file to You under the Apache License, Version 2.0
457  * (the "License"); you may not use this file except in compliance with
458  * the License.  You may obtain a copy of the License at
459  *
460  *      http://www.apache.org/licenses/LICENSE-2.0
461  *
462  * Unless required by applicable law or agreed to in writing, software
463  * distributed under the License is distributed on an "AS IS" BASIS,
464  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
465  * See the License for the specific language governing permissions and
466  * limitations under the License.
467  */
468 
469 /*
470  * $Id: SAX2CountHandlers.hpp 679377 2008-07-24 11:56:42Z borisk $
471  */
472 
473 
474 // ---------------------------------------------------------------------------
475 //  Includes
476 // ---------------------------------------------------------------------------
477 #include <xercesc/sax2/Attributes.hpp>
```

```
478 #include <xercesc/sax2/DefaultHandler.hpp>
479
480 XERCES_CPP_NAMESPACE_USE
481
482 class SAX2CountHandlers : public DefaultHandler
483 {
484 public:
485     // -----------------------------------------------------------------------
486     //  Constructors and Destructor
487     // -----------------------------------------------------------------------
488     SAX2CountHandlers();
489     ~SAX2CountHandlers();
490
491
492     // -----------------------------------------------------------------------
493     //  Getter methods
494     // -----------------------------------------------------------------------
495     XMLSize_t getElementCount() const
496     {
497         return fElementCount;
498     }
499
500     XMLSize_t getAttrCount() const
501     {
502         return fAttrCount;
503     }
504
505     XMLSize_t getCharacterCount() const
506     {
507         return fCharacterCount;
508     }
509
510     bool getSawErrors() const
511     {
512         return fSawErrors;
513     }
514
515     XMLSize_t getSpaceCount() const
516     {
517         return fSpaceCount;
518     }
519
520
521     // -----------------------------------------------------------------------
522     //  Handlers for the SAX ContentHandler interface
523     // -----------------------------------------------------------------------
524     void startElement(const XMLCh* const uri, const XMLCh* const localname, const XMLCh* const qname, const
525     void characters(const XMLCh* const chars, const XMLSize_t length);
526     void ignorableWhitespace(const XMLCh* const chars, const XMLSize_t length);
527     void startDocument();
528
529
530     // -----------------------------------------------------------------------
531     //  Handlers for the SAX ErrorHandler interface
532     // -----------------------------------------------------------------------
533   void warning(const SAXParseException& exc);
534     void error(const SAXParseException& exc);
535     void fatalError(const SAXParseException& exc);
536     void resetErrors();
537
538
539 private:
540     // -----------------------------------------------------------------------
541     //  Private data members
542     //
543     //  fAttrCount
544     //  fCharacterCount
545     //  fElementCount
546     //  fSpaceCount
547     //      These are just counters that are run upwards based on the input
548     //      from the document handlers.
```

```
549     //
550     //  fSawErrors
551     //      This is set by the error handlers, and is queryable later to
552     //      see if any errors occured.
553     // -----------------------------------------------------------------------
554     XMLSize_t       fAttrCount;
555     XMLSize_t       fCharacterCount;
556     XMLSize_t       fElementCount;
557     XMLSize_t       fSpaceCount;
558     bool            fSawErrors;
559 };
560
561
562 ============================================================================
563 SAX2CountHandlers.cpp
564 ============================================================================
565
566
567 /*
568  * Licensed to the Apache Software Foundation (ASF) under one or more
569  * contributor license agreements.  See the NOTICE file distributed with
570  * this work for additional information regarding copyright ownership.
571  * The ASF licenses this file to You under the Apache License, Version 2.0
572  * (the "License"); you may not use this file except in compliance with
573  * the License.  You may obtain a copy of the License at
574  *
575  *      http://www.apache.org/licenses/LICENSE-2.0
576  *
577  * Unless required by applicable law or agreed to in writing, software
578  * distributed under the License is distributed on an "AS IS" BASIS,
579  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
580  * See the License for the specific language governing permissions and
581  * limitations under the License.
582  */
583
584 /*
585  * $Id: SAX2CountHandlers.cpp 557282 2007-07-18 14:54:15Z amassari $
586  */
587
588 // ---------------------------------------------------------------------------
589 //  Includes
590 // ---------------------------------------------------------------------------
591 #include "SAX2Count.hpp"
592 #include <xercesc/sax2/Attributes.hpp>
593 #include <xercesc/sax/SAXParseException.hpp>
594 #include <xercesc/sax/SAXException.hpp>
595
596
597
598 // ---------------------------------------------------------------------------
599 //  SAX2CountHandlers: Constructors and Destructor
600 // ---------------------------------------------------------------------------
601 SAX2CountHandlers::SAX2CountHandlers() :
602
603     fAttrCount(0)
604     , fCharacterCount(0)
605     , fElementCount(0)
606     , fSpaceCount(0)
607     , fSawErrors(false)
608 {
609 }
610
611 SAX2CountHandlers::~SAX2CountHandlers()
612 {
613 }
614
```

```
615 // ---------------------------------------------------------------------------
616 //  SAX2CountHandlers: Implementation of the SAX DocumentHandler interface
617 // ---------------------------------------------------------------------------
618 void SAX2CountHandlers::startElement(const XMLCh* const /* uri */
619                                      , const XMLCh* const /* localname */
620                                      , const XMLCh* const /* qname */
621                                      , const Attributes& attrs)
622 {
623     fElementCount++;
624     fAttrCount += attrs.getLength();
625 }
626
627 void SAX2CountHandlers::characters(  const   XMLCh* const   /* chars */
628                       , const XMLSize_t length)
629 {
630     fCharacterCount += length;
631 }
632
633 void SAX2CountHandlers::ignorableWhitespace( const   XMLCh* const /* chars */
634                           , const XMLSize_t length)
635 {
636     fSpaceCount += length;
637 }
638
639 void SAX2CountHandlers::startDocument()
640 {
641     fAttrCount = 0;
642     fCharacterCount = 0;
643     fElementCount = 0;
644     fSpaceCount = 0;
645 }
646
647
648 // ---------------------------------------------------------------------------
649 //  SAX2CountHandlers: Overrides of the SAX ErrorHandler interface
650 // ---------------------------------------------------------------------------
651 void SAX2CountHandlers::error(const SAXParseException& e)
652 {
653     fSawErrors = true;
654     XERCES_STD_QUALIFIER cerr << "\nError at file " << StrX(e.getSystemId())
655      << ", line " << e.getLineNumber()
656      << ", char " << e.getColumnNumber()
657          << "\n  Message: " << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
658 }
659
660 void SAX2CountHandlers::fatalError(const SAXParseException& e)
661 {
662     fSawErrors = true;
663     XERCES_STD_QUALIFIER cerr << "\nFatal Error at file " << StrX(e.getSystemId())
664      << ", line " << e.getLineNumber()
665      << ", char " << e.getColumnNumber()
666          << "\n  Message: " << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
667 }
668
669 void SAX2CountHandlers::warning(const SAXParseException& e)
670 {
671     XERCES_STD_QUALIFIER cerr << "\nWarning at file " << StrX(e.getSystemId())
672      << ", line " << e.getLineNumber()
673      << ", char " << e.getColumnNumber()
674          << "\n  Message: " << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
675 }
676
677 void SAX2CountHandlers::resetErrors()
678 {
679     fSawErrors = false;
680 }
681
682
683 ============================================================================
```