```
 1 /*
 2  * Licensed to the Apache Software Foundation (ASF) under one or more
 3  * contributor license agreements.  See the NOTICE file distributed with
 4  * this work for additional information regarding copyright ownership.
 5  * The ASF licenses this file to You under the Apache License, Version 2.0
 6  * (the "License"); you may not use this file except in compliance with
 7  * the License.  You may obtain a copy of the License at
 8  *
 9  *      http://www.apache.org/licenses/LICENSE-2.0
10  *
11  * Unless required by applicable law or agreed to in writing, software
12  * distributed under the License is distributed on an "AS IS" BASIS,
13  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14  * See the License for the specific language governing permissions and
15  * limitations under the License.
16  */
17
18 /*
19  * $Id: SAXCount.hpp 471735 2006-11-06 13:53:58Z amassari $
20  */
21
22
23 // ---------------------------------------------------------------------------
24 //  Includes for all the program files to see
25 // ---------------------------------------------------------------------------
26 #include <xercesc/util/PlatformUtils.hpp>
27 #include <stdlib.h>
28 #include <string.h>
29 #if defined(XERCES_NEW_IOSTREAMS)
30 #include <iostream>
31 #else
32 #include <iostream.h>
33 #endif
34 #include <xercesc/parsers/SAXParser.hpp>
35 #include "SAXCountHandlers.hpp"
36
37
38 // ---------------------------------------------------------------------------
39 //  This is a simple class that lets us do easy (though not terribly efficient)
40 //  trancoding of XMLCh data to local code page for display.
41 // ---------------------------------------------------------------------------
42 class StrX
43 {
44 public :
45     // -----------------------------------------------------------------------
46     //  Constructors and Destructor
47     // -----------------------------------------------------------------------
48     StrX(const XMLCh* const toTranscode)
49     {
50         // Call the private transcoding method
51         fLocalForm = XMLString::transcode(toTranscode);
52     }
53
54     ~StrX()
55     {
56         XMLString::release(&fLocalForm);
57     }
58
59     // -----------------------------------------------------------------------
60     //  Getter methods
61     // -----------------------------------------------------------------------
62     const char* localForm() const
```

```
63      {
64          return fLocalForm;
65      }
66
67  private :
68      // -----------------------------------------------------------------------
69      //  Private data members
70      //
71      //  fLocalForm
72      //      This is the local code page form of the string.
73      // -----------------------------------------------------------------------
74      char*   fLocalForm;
75  };
76
77  inline XERCES_STD_QUALIFIER ostream& operator<<(XERCES_STD_QUALIFIER ostream& target, const
StrX& toDump)
78  {
79      target << toDump.localForm();
80      return target;
81  }
```

```
 1 /*
 2  * Licensed to the Apache Software Foundation (ASF) under one or more
 3  * contributor license agreements.  See the NOTICE file distributed with
 4  * this work for additional information regarding copyright ownership.
 5  * The ASF licenses this file to You under the Apache License, Version 2.0
 6  * (the "License"); you may not use this file except in compliance with
 7  * the License.  You may obtain a copy of the License at
 8  *
 9  *      http://www.apache.org/licenses/LICENSE-2.0
10  *
11  * Unless required by applicable law or agreed to in writing, software
12  * distributed under the License is distributed on an "AS IS" BASIS,
13  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14  * See the License for the specific language governing permissions and
15  * limitations under the License.
16  */
17
18 /*
19  * $Id: SAXCount.cpp 471735 2006-11-06 13:53:58Z amassari $
20  */
21
22
23 // ---------------------------------------------------------------------------
24 //  Includes
25 // ---------------------------------------------------------------------------
26 #include "SAXCount.hpp"
27 #if defined(XERCES_NEW_IOSTREAMS)
28 #include <fstream>
29 #else
30 #include <fstream.h>
31 #endif
32 #include <xercesc/util/OutOfMemoryException.hpp>
33
34
35 // ---------------------------------------------------------------------------
36 //  Local helper methods
37 // ---------------------------------------------------------------------------
38 void usage()
39 {
40     XERCES_STD_QUALIFIER cout << "\nUsage:\n"
41             "    SAXCount [options] <XML file | List file>\n\n"
42             "This program invokes the SAX Parser, and then prints the\n"
43             "number of elements, attributes, spaces and characters found\n"
44             "in each XML file, using SAX API.\n\n"
45             "Options:\n"
46             "    -l          Indicate the input file is a List File that has a list of xml
files.\n"
47             "                Default to off (Input file is an XML file).\n"
48             "    -v=xxx      Validation scheme [always | never | auto*].\n"
49             "    -n          Enable namespace processing. Defaults to off.\n"
50             "    -s          Enable schema processing. Defaults to off.\n"
51             "    -f          Enable full schema constraint checking. Defaults to off.\n"
52             "    -locale=ll_CC specify the locale, default: en_US.\n"
53     "    -?          Show this help.\n\n"
54             "  * = Default if not provided explicitly.\n"
55          << XERCES_STD_QUALIFIER endl;
56 }
57
58
59 // ---------------------------------------------------------------------------
60 //  Program entry point
61 // ---------------------------------------------------------------------------
```

```
 62 int main(int argC, char* argV[])
 63 {
 64
 65     // Check command line and extract arguments.
 66     if (argC < 2)
 67     {
 68         usage();
 69         return 1;
 70     }
 71
 72     const char*             xmlFile = 0;
 73     SAXParser::ValSchemes   valScheme = SAXParser::Val_Auto;
 74     bool                    doNamespaces       = false;
 75     bool                    doSchema           = false;
 76     bool                    schemaFullChecking = false;
 77     bool                    doList = false;
 78     bool                    errorOccurred = false;
 79     bool                    recognizeNEL = false;
 80     char                    localeStr[64];
 81     memset(localeStr, 0, sizeof localeStr);
 82
 83     int argInd;
 84     for (argInd = 1; argInd < argC; argInd++)
 85     {
 86         // Break out on first parm not starting with a dash
 87         if (argV[argInd][0] != '-')
 88             break;
 89
 90         // Watch for special case help request
 91         if (!strcmp(argV[argInd], "-?"))
 92         {
 93             usage();
 94             return 2;
 95         }
 96          else if (!strncmp(argV[argInd], "-v=", 3)
 97               ||  !strncmp(argV[argInd], "-V=", 3))
 98         {
 99             const char* const parm = &argV[argInd][3];
100
101             if (!strcmp(parm, "never"))
102                 valScheme = SAXParser::Val_Never;
103             else if (!strcmp(parm, "auto"))
104                 valScheme = SAXParser::Val_Auto;
105             else if (!strcmp(parm, "always"))
106                 valScheme = SAXParser::Val_Always;
107             else
108             {
109                 XERCES_STD_QUALIFIER cerr << "Unknown -v= value: " << parm <<
XERCES_STD_QUALIFIER endl;
110                 return 2;
111             }
112         }
113          else if (!strcmp(argV[argInd], "-n")
114               ||  !strcmp(argV[argInd], "-N"))
115         {
116             doNamespaces = true;
117         }
118          else if (!strcmp(argV[argInd], "-s")
119               ||  !strcmp(argV[argInd], "-S"))
120         {
121             doSchema = true;
122         }
```

```
123          else if (!strcmp(argV[argInd], "-f")
124               ||  !strcmp(argV[argInd], "-F"))
125         {
126             schemaFullChecking = true;
127         }
128          else if (!strcmp(argV[argInd], "-l")
129               ||  !strcmp(argV[argInd], "-L"))
130         {
131             doList = true;
132         }
133          else if (!strcmp(argV[argInd], "-special:nel"))
134         {
135             // turning this on will lead to non-standard compliance behaviour
136             // it will recognize the unicode character 0x85 as new line character
137             // instead of regular character as specified in XML 1.0
138             // do not turn this on unless really necessary
139              recognizeNEL = true;
140         }
141          else if (!strncmp(argV[argInd], "-locale=", 8))
142         {
143              // Get out the end of line
144              strcpy(localeStr, &(argV[argInd][8]));
145         }
146         else
147         {
148             XERCES_STD_QUALIFIER cerr << "Unknown option '" << argV[argInd]
149                 << "', ignoring it\n" << XERCES_STD_QUALIFIER endl;
150         }
151     }
152
153     //
154     //  There should at least one parameter left, and that
155     //  should be the file name(s).
156     //
157     if (argInd == argC)
158     {
159         usage();
160         return 1;
161     }
162
163     // Initialize the XML4C2 system
164     try
165     {
166         if (strlen(localeStr))
167         {
168             XMLPlatformUtils::Initialize(localeStr);
169         }
170         else
171         {
172             XMLPlatformUtils::Initialize();
173         }
174
175         if (recognizeNEL)
176         {
177             XMLPlatformUtils::recognizeNEL(recognizeNEL);
178         }
179     }
180
181     catch (const XMLException& toCatch)
182     {
183         XERCES_STD_QUALIFIER cerr << "Error during initialization! Message:\n"
184             << StrX(toCatch.getMessage()) << XERCES_STD_QUALIFIER endl;
```

```
185         return 1;
186     }
187
188     //
189     //  Create a SAX parser object. Then, according to what we were told on
190     //  the command line, set it to validate or not.
191     //
192     SAXParser* parser = new SAXParser;
193
194     parser->setValidationScheme(valScheme);
195     parser->setDoNamespaces(doNamespaces);
196     parser->setDoSchema(doSchema);
197     parser->setValidationSchemaFullChecking(schemaFullChecking);
198
199     //
200     //  Create our SAX handler object and install it on the parser, as the
201     //  document and error handler.
202     //
203     SAXCountHandlers handler;
204     parser->setDocumentHandler(&handler);
205     parser->setErrorHandler(&handler);
206
207
208     //
209     //  Get the starting time and kick off the parse of the indicated
210     //  file. Catch any exceptions that might propogate out of it.
211     //
212     unsigned long duration;
213
214     XERCES_STD_QUALIFIER ifstream fin;
215
216     // the input is a list file
217     if (doList)
218         fin.open(argV[argInd]);
219
220     if (fin.fail()) {
221         XERCES_STD_QUALIFIER cerr <<"Cannot open the list file: " << argV[argInd] <<
XERCES_STD_QUALIFIER endl;
222         return 2;
223     }
224
225     while (true)
226     {
227         char fURI[1000];
228         //initialize the array to zeros
229         memset(fURI,0,sizeof(fURI));
230
231         if (doList) {
232             if (! fin.eof() ) {
233                 fin.getline (fURI, sizeof(fURI));
234                 if (!*fURI)
235                     continue;
236                 else {
237                     xmlFile = fURI;
238                     XERCES_STD_QUALIFIER cerr << "==Parsing== " << xmlFile <<
XERCES_STD_QUALIFIER endl;
239                 }
240             }
241             else
242                 break;
243         }
244         else {
```

```
245             if (argInd < argC)
246             {
247                  xmlFile = argV[argInd];
248                  argInd++;
249             }
250             else
251                 break;
252         }
253
254         //reset error count first
255         handler.resetErrors();
256
257         try
258         {
259             const unsigned long startMillis = XMLPlatformUtils::getCurrentMillis();
260             parser->parse(xmlFile);
261             const unsigned long endMillis = XMLPlatformUtils::getCurrentMillis();
262             duration = endMillis - startMillis;
263         }
264         catch (const OutOfMemoryException&)
265         {
266             XERCES_STD_QUALIFIER cerr << "OutOfMemoryException" << XERCES_STD_QUALIFIER endl;
267             errorOccurred = true;
268             continue;
269         }
270         catch (const XMLException& e)
271         {
272             XERCES_STD_QUALIFIER cerr << "\nError during parsing: '" << xmlFile << "'\n"
273                 << "Exception message is:  \n"
274                 << StrX(e.getMessage()) << "\n" << XERCES_STD_QUALIFIER endl;
275             errorOccurred = true;
276             continue;
277         }
278
279         catch (...)
280         {
281             XERCES_STD_QUALIFIER cerr << "\nUnexpected exception during parsing: '" <<
xmlFile << "'\n";
282             errorOccurred = true;
283             continue;
284         }
285
286
287         // Print out the stats that we collected and time taken
288         if (!handler.getSawErrors())
289         {
290             XERCES_STD_QUALIFIER cout << xmlFile << ": " << duration << " ms ("
291                 << handler.getElementCount() << " elems, "
292                 << handler.getAttrCount() << " attrs, "
293                 << handler.getSpaceCount() << " spaces, "
294                 << handler.getCharacterCount() << " chars)" << XERCES_STD_QUALIFIER endl;
295         }
296         else
297             errorOccurred = true;
298     }
299
300     if (doList)
301         fin.close();
302
303     //
304     //  Delete the parser itself.  Must be done prior to calling Terminate, below.
305     //
```

```
306     delete parser;
307
308     // And call the termination method
309     XMLPlatformUtils::Terminate();
310
311     if (errorOccurred)
312         return 4;
313     else
314         return 0;
315
316 }
```

```
 1 /*
 2  * Licensed to the Apache Software Foundation (ASF) under one or more
 3  * contributor license agreements.  See the NOTICE file distributed with
 4  * this work for additional information regarding copyright ownership.
 5  * The ASF licenses this file to You under the Apache License, Version 2.0
 6  * (the "License"); you may not use this file except in compliance with
 7  * the License.  You may obtain a copy of the License at
 8  *
 9  *      http://www.apache.org/licenses/LICENSE-2.0
10  *
11  * Unless required by applicable law or agreed to in writing, software
12  * distributed under the License is distributed on an "AS IS" BASIS,
13  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14  * See the License for the specific language governing permissions and
15  * limitations under the License.
16  */
17
18 /*
19  * $Id: SAXCountHandlers.hpp 679377 2008-07-24 11:56:42Z borisk $
20  */
21
22
23 // ---------------------------------------------------------------------------
24 //  Includes
25 // ---------------------------------------------------------------------------
26 #include <xercesc/sax/HandlerBase.hpp>
27
28 XERCES_CPP_NAMESPACE_USE
29
30 XERCES_CPP_NAMESPACE_BEGIN
31 class AttributeList;
32 XERCES_CPP_NAMESPACE_END
33
34 class SAXCountHandlers : public HandlerBase
35 {
36 public:
37     // -----------------------------------------------------------------------
38     //  Constructors and Destructor
39     // -----------------------------------------------------------------------
40     SAXCountHandlers();
41     ~SAXCountHandlers();
42
43
44     // -----------------------------------------------------------------------
45     //  Getter methods
46     // -----------------------------------------------------------------------
47     XMLSize_t getElementCount() const
48     {
49         return fElementCount;
50     }
51
52     XMLSize_t getAttrCount() const
53     {
54         return fAttrCount;
55     }
56
57     XMLSize_t getCharacterCount() const
58     {
59         return fCharacterCount;
60     }
61
62     bool getSawErrors() const
```

```
 63     {
 64         return fSawErrors;
 65     }
 66
 67     XMLSize_t getSpaceCount() const
 68     {
 69         return fSpaceCount;
 70     }
 71
 72
 73     // -----------------------------------------------------------------------
 74     //  Handlers for the SAX DocumentHandler interface
 75     // -----------------------------------------------------------------------
 76 void startElement(const XMLCh* const name, AttributeList& attributes);
 77     void characters(const XMLCh* const chars, const XMLSize_t length);
 78     void ignorableWhitespace(const XMLCh* const chars, const XMLSize_t length);
 79     void resetDocument();
 80
 81
 82     // -----------------------------------------------------------------------
 83     //  Handlers for the SAX ErrorHandler interface
 84     // -----------------------------------------------------------------------
 85 void warning(const SAXParseException& exc);
 86     void error(const SAXParseException& exc);
 87     void fatalError(const SAXParseException& exc);
 88     void resetErrors();
 89
 90
 91 private:
 92     // -----------------------------------------------------------------------
 93     //  Private data members
 94     //
 95     //  fAttrCount
 96     //  fCharacterCount
 97     //  fElementCount
 98     //  fSpaceCount
 99     //      These are just counters that are run upwards based on the input
100     //      from the document handlers.
101     //
102     //  fSawErrors
103     //      This is set by the error handlers, and is queryable later to
104     //      see if any errors occured.
105     // -----------------------------------------------------------------------
106     XMLSize_t       fAttrCount;
107     XMLSize_t       fCharacterCount;
108     XMLSize_t       fElementCount;
109     XMLSize_t       fSpaceCount;
110     bool            fSawErrors;
111 };
```

```
 1 /*
 2  * Licensed to the Apache Software Foundation (ASF) under one or more
 3  * contributor license agreements.  See the NOTICE file distributed with
 4  * this work for additional information regarding copyright ownership.
 5  * The ASF licenses this file to You under the Apache License, Version 2.0
 6  * (the "License"); you may not use this file except in compliance with
 7  * the License.  You may obtain a copy of the License at
 8  *
 9  *      http://www.apache.org/licenses/LICENSE-2.0
10  *
11  * Unless required by applicable law or agreed to in writing, software
12  * distributed under the License is distributed on an "AS IS" BASIS,
13  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14  * See the License for the specific language governing permissions and
15  * limitations under the License.
16  */
17
18 /*
19  * $Id: SAXCountHandlers.cpp 557282 2007-07-18 14:54:15Z amassari $
20  */
21
22 // ---------------------------------------------------------------------------
23 //  Includes
24 // ---------------------------------------------------------------------------
25 #include "SAXCount.hpp"
26 #include <xercesc/sax/AttributeList.hpp>
27 #include <xercesc/sax/SAXParseException.hpp>
28 #include <xercesc/sax/SAXException.hpp>
29
30
31 using namespace xercesc ;
32
33
34 // ---------------------------------------------------------------------------
35 //  SAXCountHandlers: Constructors and Destructor
36 // ---------------------------------------------------------------------------
37 SAXCountHandlers::SAXCountHandlers() :
38
39     fAttrCount(0)
40     , fCharacterCount(0)
41     , fElementCount(0)
42     , fSpaceCount(0)
43     , fSawErrors(false)
44 {
45 }
46
47 SAXCountHandlers::~SAXCountHandlers()
48 {
49 }
50
51
52 // ---------------------------------------------------------------------------
53 //  SAXCountHandlers: Implementation of the SAX DocumentHandler interface
54 // ---------------------------------------------------------------------------
55 void SAXCountHandlers::startElement(const   XMLCh* const    /* name */
56                                     ,       AttributeList&  attributes)
57 {
58     fElementCount++;
59     fAttrCount += attributes.getLength();
60 }
61
62 void SAXCountHandlers::characters(  const   XMLCh* const    /* chars */
```

```
 63     , const XMLSize_t       length)
 64 {
 65     fCharacterCount += length;
 66 }
 67
 68 void SAXCountHandlers::ignorableWhitespace( const   XMLCh* const /* chars */
 69     , const XMLSize_t    length)
 70 {
 71     fSpaceCount += length;
 72 }
 73
 74 void SAXCountHandlers::resetDocument()
 75 {
 76     fAttrCount = 0;
 77     fCharacterCount = 0;
 78     fElementCount = 0;
 79     fSpaceCount = 0;
 80 }
 81
 82
 83 // ---------------------------------------------------------------------------
 84 //  SAXCountHandlers: Overrides of the SAX ErrorHandler interface
 85 // ---------------------------------------------------------------------------
 86 void SAXCountHandlers::error(const SAXParseException& e)
 87 {
 88     fSawErrors = true;
 89     XERCES_STD_QUALIFIER cerr << "\nError at file " << StrX(e.getSystemId())
 90  << ", line " << e.getLineNumber()
 91  << ", char " << e.getColumnNumber()
 92          << "\n  Message: " << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
 93 }
 94
 95 void SAXCountHandlers::fatalError(const SAXParseException& e)
 96 {
 97     fSawErrors = true;
 98     XERCES_STD_QUALIFIER cerr << "\nFatal Error at file " << StrX(e.getSystemId())
 99  << ", line " << e.getLineNumber()
100  << ", char " << e.getColumnNumber()
101          << "\n  Message: " << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
102 }
103
104 void SAXCountHandlers::warning(const SAXParseException& e)
105 {
106     XERCES_STD_QUALIFIER cerr << "\nWarning at file " << StrX(e.getSystemId())
107  << ", line " << e.getLineNumber()
108  << ", char " << e.getColumnNumber()
109          << "\n  Message: " << StrX(e.getMessage()) << XERCES_STD_QUALIFIER endl;
110 }
111
112 void SAXCountHandlers::resetErrors()
113 {
114     fSawErrors = false;
115 }
```