Symbian XML parsing plugins

CParser default plugin - Expat

`CParser` class is using by default Expat parser. It is stream based let's say SAX-like parser. Symbian implementation of the Expat XML parser plugin is a little bit old (S^3 has version 1.95.5 from 2002 year) and it has one known problem - it wrongly parses XML attributes of size larger tham 12kB. Elements with such large attributes are notified to the application as a content of the parent element and also they missing first ~12kB of data.

Source code of the Expat parser plugin can be found in Symbian Foundation source code:
\PDK_4.0.a\src_oss_os.zip\src_oss_os_xmlsrv.zip\sf\os\xmlsrv\xml\xmlexpatparser
Symbian Foundation source code is available on SourceForge prtal:
http://sourceforge.net/projects/symbiandump/files/PDK_4.0.a/

CParser Libxml2 plugin

To resolve problem with large attributes and parse XML by `CParse` class, we can change XML parsing plugin from Expat to Libxml2. It is available at least from S^3 (tested on Nokia E7). To select parsing plugin we are passing it during construction of `CParse` class:

```
_LIT8(KMimeType, "text/xml");
_LIT8(KParser, "libxml2");
iParserPlugin = CMatchData::NewL();
iParserPlugin->SetMimeTypeL( KMimeType() );
iParserPlugin->SetVariantL( KParser() );
iParser = CParser::NewL( *iParserPlugin, *this ); // assuming the this class implements
MContentHandler interface
```

This will resolve large attribute problem without need of moving to DOM parser and with only little code change.
Remember to destroy CMatchData in your class destructor.

Source code of the Libxml2 parser plugin can be found also in Symbian Foundation source code:
\PDK_4.0.a\src_oss_os.zip\src_oss_os_xmlsrv.zip\sf\os\xmlsrv\xml\xmllibxml2parser

CParser optimization using string table

Using SAX parsing element and attributes resolve requires comparison of the element and attributes names. To minimize string comparisons count we can pass predefined string table to the `CParser` object and get elements and attributes indexes in our table.

Firstly we need to declare our string table:

```
#include <stringpool.h>

_LIT8(KParserStr_1, "element1");
_LIT8(KParserStr_2, "element2");
_LIT8(KParserStr_3, "attr1");
_LIT8(KParserStr_4, "attr2");

const void * const KParserStringPointers[] =
{
    (const void*)&KParserStr_1,
    (const void*)&KParserStr_2,
    (const void*)&KParserStr_3,
    (const void*)&KParserStr_4
};
```


(C) Copyright Nokia 2014. All rights reserved.

```
const TStringTable KParserStringTable = { 4, KParserStringPointers, ETrue };   // to
use it with CParser case sensitive must be set to ETrue

enum TParserStringTable
    {
    EParserString_element1,
    EParserString_element2,
    EParserString_attr1,
    EParserString_attr2
    };
```

Now we need to load the table to the parser. We can do it just before starting of parsing:

```
iParser->StringPool().OpenL( KParserStringTable );
iParser->ParseBeginL();
...
```

Now in callbacks we can resolve element and attribute string names just by enumeration:

```
void CMyParser::OnStartElementL(const RTagInfo& aElement, const RAttributeArray&
aAttributes,TInt aErrorCode)
    {
    TInt idx = aElement.LocalName().Index( KParserStringTable );
    switch ( idx )
        {
        case EParserString_element1:
                ParseElement( aAttributes );
            break;
        case EParserString_element1:
                ParseElement( aAttributes );
            break;
        default:
            break;
        }
    }

void CMyParser::ParseElement(const RAttributeArray& aAttributes)
    {
    for ( TInt i = 0; i < aAttributes.Count(); i++ )
        {
        TInt idx = aAttributes[ i ].Attribute().LocalName().Index( KParserStringTable );
        switch ( idx )
            {
            case EParserString_attr1:
                break;
            case EParserString_attr2:
                break;
            default:
                break;
            }
        }
    }
```


(C) Copyright Nokia 2014. All rights reserved.

Version Hint

Windows Phone: [[Category:Windows Phone]]
[[Category:Windows Phone 7.5]]
[[Category:Windows Phone 8]]

Nokia Asha: [[Category:Nokia Asha]]
[[Category:Nokia Asha Platform 1.0]]

Series 40: [[Category:Series 40]]
[[Category:Series 40 1st Edition]] [[Category:Series 40 2nd Edition]]
[[Category:Series 40 3rd Edition (initial release)]] [[Category:Series 40 3rd Edition FP1]] [[Category:Series 40 3rd Edition FP2]]
[[Category:Series 40 5th Edition (initial release)]] [[Category:Series 40 5th Edition FP1]]
[[Category:Series 40 6th Edition (initial release)]] [[Category:Series 40 6th Edition FP1]] [[Category:Series 40 Developer Platform 1.0]] [[Category:Series 40 Developer Platform 1.1]] [[Category:Series 40 Developer Platform 2.0]]

Symbian: [[Category:Symbian]]
[[Category:S60 1st Edition]] [[Category:S60 2nd Edition (initial release)]] [[Category:S60 2nd Edition FP1]] [[Category:S60 2nd Edition FP2]] [[Category:S60 2nd Edition FP3]]
[[Category:S60 3rd Edition (initial release)]] [[Category:S60 3rd Edition FP1]] [[Category:S60 3rd Edition FP2]]
[[Category:S60 5th Edition]]
[[Category:Symbian^3]] [[Category:Symbian Anna]] [[Category:Nokia Belle]]

(C) Copyright Nokia 2014. All rights reserved.