Subject: CParser: proposal of new functions
Posted by omari on Wed, 16 May 2012 16:40:32 GMT
View Forum Message <> Reply to Message

Hello ,

I have to load data in a Vector<TestCase>
the struct of the data file is:

TEST CASE:
Name of Test case: TC_0000
Test case version: 1
Author:
Date: 10/05/2012
Test type: nominal
SRD_REQUIREMENT: SRD-REQ-0628
SRD version: G02
Precondition:
...
Description:
visit LoginPage
enter userID
....
Expected Result:
...
END OF TEST CASE

to do that i use CParser and i have added a couple of function:

in CParser.h:

```
bool IsId2(const char *s1, const char *s2) const;
bool IsId3(const char *s1, const char *s2, const char *s3) const;
bool Id2(const char *s1, const char *s2);
bool Id3(const char *s1, const char *s2, const char *s3);
String ReadId2() throw(Error);
String ReadId3() throw(Error);
String ReadUntil(int delim);
String ReadLine() {return ReadUntil('\n');}
```

the function ReadUntil is based on ReadOneString()

if you find this functions useful, i would be glad if you add them to CParser..

in CParser.cpp:

```
bool CParser::IsId2(const char *s1, const char *s2) const
{
 CParser p = *this;

 if (!p.Id(s1)) return false;

 return p.IsId(s2);
}

bool CParser::IsId3(const char *s1, const char *s2, const char *s3) const
{

 CParser p = *this;

 if (!p.Id(s1)) return false;
 if (!p.Id(s2)) return false;

 return p.IsId(s3);
}

bool CParser::Id2(const char *s1, const char *s2)
{
 CParser p = *this;

 if (!p.Id(s1)) return false;
 if (!p.Id(s2)) return false;

 *this = p;

 return true;
}

bool CParser::Id3(const char *s1, const char *s2, const char *s3)
{
 CParser p = *this;

 if (!p.Id(s1)) return false;
 if (!p.Id(s2)) return false;
 if (!p.Id(s3)) return false;

 *this = p;

 return true;
}
```

```
String CParser::ReadId2() throw(Error)
{
 String ret ="";

 ret << ReadId();
 const char* p = GetSpacePtr();
 ret << String(p, term - p);
 ret << ReadId();

 return ret;
}

String CParser::ReadId3() throw(Error)
{
 String ret ="";

 ret << ReadId();
 const char* p = GetSpacePtr();
 ret << String(p, term - p);
 ret << ReadId();
 p = GetSpacePtr();
 ret << String(p, term - p);
 ret << ReadId();

 return ret;
}

String CParser::ReadUntil(int delim)
{
 StringBuffer result;

 for(;;) {
  if(*term == delim) {
   term++;
   DoSpaces();
   return result;
  }
  else
  if(*term == '\\') {
   switch(*++term) {
   case 'a': result.Cat('\a'); term++; break;
   case 'b': result.Cat('\b'); term++; break;
   case 't': result.Cat('\t'); term++; break;
   case 'v': result.Cat('\v'); term++; break;
   case 'n': result.Cat('\n'); term++; break;
   case 'r': result.Cat('\r'); term++; break;
   case 'f': result.Cat('\f'); term++; break;
   case 'x': {
```

```
  int hex = 0;
  if(IsXDigit(*++term)) {
   hex = ctoi(*term);
   if(IsXDigit(*++term)) {
    hex = 16 * hex + (*term >= 'A' ? ToUpper(*term) - 'A' + 10 : *term - '0');
    term++;
   }
  }
  result.Cat(hex);
  break;
 }
 case 'u':
  if(uescape) {
   int hex = 0;
   if(IsXDigit(*++term)) {
    hex = ctoi(*term);
    if(IsXDigit(*++term)) {
     hex = 16 * hex + (*term >= 'A' ? ToUpper(*term) - 'A' + 10 : *term - '0');
     if(IsXDigit(*++term)) {
      hex = 16 * hex + (*term >= 'A' ? ToUpper(*term) - 'A' + 10 : *term - '0');
      if(IsXDigit(*++term)) {
       hex = 16 * hex + (*term >= 'A' ? ToUpper(*term) - 'A' + 10 : *term - '0');
       term++;
      }
     }
    }
   }
   result.Cat(WString(hex, 1).ToString());
  }
  else
   result.Cat(*term++);
  break;
 default:
  if(*term >= '0' && *term <= '7') {
   int oct = *term++ - '0';
   if(*term >= '0' && *term <= '7')
    oct = 8 * oct + *term++ - '0';
   if(*term >= '0' && *term <= '7')
    oct = 8 * oct + *term++ - '0';
   result.Cat(oct);
  }
  else
   result.Cat(*term++);
  break;
 }
}
else {
 if(*term == '\0')
```

```
    return result;
   result.Cat(*term++);
  }
 }

 DoSpaces();
 return result;
}
```

Thanks

Subject: Re: CParser: proposal of new functions
Posted by Sender Ghost on Wed, 16 May 2012 19:08:14 GMT
View Forum Message <> Reply to Message

Hello, omari.

I think, you could do the same without changing the CParser class directly:

```
#include <Core/Core.h>

using namespace Upp;

class OmariCParser : public CParser {
public:
 bool IsId2(const char *s1, const char *s2);
 bool IsId3(const char *s1, const char *s2, const char *s3);
 bool Id2(const char *s1, const char *s2);
 bool Id3(const char *s1, const char *s2, const char *s3);
 String ReadId2() throw(Error);
 String ReadId3() throw(Error);
 String ReadUntil(int delim);
 String ReadLine() { return ReadUntil('\n'); }

 OmariCParser(const char *ptr) : CParser(ptr) { }
 OmariCParser(const char *ptr, const char *fn, int line = 1) : CParser(ptr, fn, line) { }
};

bool OmariCParser::IsId2(const char *s1, const char *s2)
{
 if (!Id(s1)) return false;
 return IsId(s2);
}

bool OmariCParser::IsId3(const char *s1, const char *s2, const char *s3)
```

```
{
 if (!Id(s1) || !Id(s2)) return false;
 return IsId(s3);
}

bool OmariCParser::Id2(const char *s1, const char *s2)
{
 if (!Id(s1) || !Id(s2)) return false;
 return true;
}

bool OmariCParser::Id3(const char *s1, const char *s2, const char *s3)
{
 if (!Id(s1) || !Id(s2) || !Id(s3)) return false;
 return true;
}

String OmariCParser::ReadId2() throw(Error)
{
 String ret ="";

 ret << ReadId();
 const char* p = GetSpacePtr();
 ret << String(p, term - p);
 ret << ReadId();

 return ret;
}

String OmariCParser::ReadId3() throw(Error)
{
 String ret ="";

 ret << ReadId();
 const char* p = GetSpacePtr();
 ret << String(p, term - p);
 ret << ReadId();
 p = GetSpacePtr();
 ret << String(p, term - p);
 ret << ReadId();

 return ret;
}

String OmariCParser::ReadUntil(int delim)
{
 StringBuffer result;
```

```
for(;;) {
 if(*term == delim) {
  term++;
  DoSpaces();
  return result;
 }
 else
 if(*term == '\\') {
  switch(*++term) {
  case 'a': result.Cat('\a'); term++; break;
  case 'b': result.Cat('\b'); term++; break;
  case 't': result.Cat('\t'); term++; break;
  case 'v': result.Cat('\v'); term++; break;
  case 'n': result.Cat('\n'); term++; break;
  case 'r': result.Cat('\r'); term++; break;
  case 'f': result.Cat('\f'); term++; break;
  case 'x': {
   int hex = 0;
   if(IsXDigit(*++term)) {
    hex = ctoi(*term);
    if(IsXDigit(*++term)) {
     hex = 16 * hex + (*term >= 'A' ? ToUpper(*term) - 'A' + 10 : *term - '0');
     term++;
    }
   }
   result.Cat(hex);
   break;
  }
  case 'u':
   if(uescape) {
    int hex = 0;
    if(IsXDigit(*++term)) {
     hex = ctoi(*term);
     if(IsXDigit(*++term)) {
      hex = 16 * hex + (*term >= 'A' ? ToUpper(*term) - 'A' + 10 : *term - '0');
      if(IsXDigit(*++term)) {
       hex = 16 * hex + (*term >= 'A' ? ToUpper(*term) - 'A' + 10 : *term - '0');
       if(IsXDigit(*++term)) {
        hex = 16 * hex + (*term >= 'A' ? ToUpper(*term) - 'A' + 10 : *term - '0');
        term++;
       }
      }
     }
    }
    result.Cat(WString(hex, 1).ToString());
   }
   else
    result.Cat(*term++);
```

```
    break;
   default:
    if(*term >= '0' && *term <= '7') {
     int oct = *term++ - '0';
     if(*term >= '0' && *term <= '7')
      oct = 8 * oct + *term++ - '0';
     if(*term >= '0' && *term <= '7')
      oct = 8 * oct + *term++ - '0';
     result.Cat(oct);
    }
    else
     result.Cat(*term++);
    break;
   }
  }
  else {
   if(*term == '\0')
    return result;
   result.Cat(*term++);
  }
 }

 DoSpaces();
 return result;
}

CONSOLE_APP_MAIN
{
 const String filePath = "TestCase.txt";
 String text = LoadFile(filePath);

 if (text.IsVoid()) {
  SetExitCode(1);
  Cerr() << "Unable to load '" << filePath << "'\n";
  return;
 }

 OmariCParser p(text);
 try {
  // Parsing of text here
 }
 catch (OmariCParser::Error e) {
  Cerr() << "ERROR: " << e << '\n';
 }
}
```

The good example is Esc parser, based on CParser.

Subject: Re: CParser: proposal of new functions
Posted by cbpporter on Thu, 17 May 2012 08:04:08 GMT
View Forum Message <> Reply to Message

I'm curious what on earth were you trying to achieve with this :

```
bool CParser::Id2(const char *s1, const char *s2)
{
 CParser p = *this;

 if (!p.Id(s1)) return false;
 if (!p.Id(s2)) return false;

 *this = p;

 return true;
}
```

Subject: Re: CParser: proposal of new functions
Posted by omari on Thu, 17 May 2012 09:31:35 GMT
View Forum Message <> Reply to Message

Thank you all.

I post this message because i want to share this functions with others.

Here is an example of using functions Id2 () and ReadUntil ():

```
String name, date;

String str="request name: xxxxx \n request date: xxxxx";

CParser p(str);

if(p.Id2("request", "name"))
{
    p.PassChar(':');
    name=p.ReadUntil('\n');
}
else
if(p.Id2("request", "date"))
{
    p.PassChar(':');
    date=p.ReadUntil('\n');
}
```

and the function Id2() with a bit of comments:

```
bool CParser::Id2(const char *s1, const char *s2)
{
   // get a copy of the object
 CParser p = *this;

   // if not found s1 or s2, return false, dont change any thing in our object
 if (!p.Id(s1)) return false;
 if (!p.Id(s2)) return false;

   // if found s1 and s2, set the internal pointer of our object (term) to: after s2
 *this = p;

 return true;
}
```

Subject: Re: CParser: proposal of new functions
Posted by mirek on Sun, 20 May 2012 15:07:21 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 17 May 2012 04:04I'm curious what on earth were you trying to achieve with this :

```
bool CParser::Id2(const char *s1, const char *s2)
{
 CParser p = *this;

 if (!p.Id(s1)) return false;
 if (!p.Id(s2)) return false;

 *this = p;

 return true;
}
```

Well, see the input file...

That said, I would rather use some other method, reading the input file line by line, separating anything before ':' to get id, and only use CParser on the rest of line.

I guess the line nature of file is something that is not quite compatible with CParser.

Subject: Re: CParser: proposal of new functions
Posted by omari on Tue, 22 May 2012 08:56:32 GMT
View Forum Message <> Reply to Message

Thanks a lot Mirek.