HMM7E的专栏

个人资料

HMM7E

访问：3243次
积分：49分
排名：千里之外

原创：1篇 转载：0篇
译文：0篇 评论：10条

文章搜索

文章存档

2011年12月(1)

阅读排行

HTML解析-第二版（C/C++）(3214)

评论排行

HTML解析-第二版（C/C++）(10)

推荐文章

* 云存储的故事——元数据

最新评论

HTML解析-第二版（C/C++）
li_mountain: mark一下，有空仔细看看！^_^

HTML解析-第二版（C/C++）
feimashenhua: 谢谢lz分享！

HTML解析-第二版（C/C++）

2011-12-14 21:05 3215人阅读 评论(10) 收藏 举报

```
[cpp]
01. 背景：
02. 基于某些不着边际想法，只为取得HTML页面上的所有“URL”和“文本”，其它的内容都不在关心之列。
03. 问题：
04. 对于“文本”搜索，如果搜索了除英文以外的语言还好说些，如果要搜索的内容是英文本，
05. 那么就难以区分是“标记”还是“本文”了。对于“URL”的搜索，因为“标记”就是英文，
06. 这样就绕回到“对于‘文本’搜索”。另外字母的大小写，被转义的字符，引号，尖括号，都得处理。
07. 例如：
08. <a href="http://www.csdn.net" >csdn</a>
09. <script src="http://csdnimg.cn/xxxxxxxx.js" type="text/javascript"></script>
10. 想要搜索“csdn”这个字符串，直接以字符串遍历的法能搜索到3个，其实呢只希望搜索到1个。
11. 例如：
12. <a href="http://bbs.csdn.net" >论坛</a>
13. <a href="http://bbs.csdn.net" >论 坛</a>
14. <a href="http://bbs.csdn.net" >论  坛</a>
15. 想要搜索“论坛”这个字符串，按语义上讲，希望在搜索时能搜到3个。
16. 但直接以字符串遍历的法能搜到1个，原因在于加了“空格”后的字符串，
17. 计算机不知道对于人来讲意思并没有变。
18.
19. 总结：
20. 1:直接搜索特定字符串,不多了就是少了。
21. 2:尝试过MS的COM库，功能强大且齐全，但耗费的资源也相当多。
22. 3:耳熟能详的搜索引擎也跑过几个回合，因没有耐心翻遍所有网页只好放弃。
23.
24. 结论：
25. 只能把HTML页面完整的解析完毕才能达找到想到的东西，尽管不是全部，但情况要好很多。
26.
27. 方法：
28. HTML语句结构是：<a href="http://www.csdn.net" >aaaa</a> 或 <link href="/favicon.ico" />
29. 等等一连串类似的语句组成，并且只有嵌套没有循环（脚本只能算上面提到的“文本”）。
30. 分界符（这个词本人自己的称呼）使用的是“ <>""''=空格 ”，把两个分界符之间的内容看作一个链表节点，
31. “标记”a与“标记”/a是“父”节点与“子”节点的关系，“标记”a与“标记”href是“兄弟”节点的关系。
32. 这样的好处是不用关心“标记”含义，就可以把整个页面解析成一个二维链表。
33. 纵向可以遍历“标记”和“文本”，横向可以找到“文本”对应用“URL”。
34. 当然实际情况要复杂的多，种种异常情况都要考虑。如：转意字符，脚本中的括号对称验证等等，
35. 最糟糕是碰到错误的语法，或者根本就不是HTML页面（这个就不属性本文说明范围了）。
36.
37.
38.
39. ////////////////////////////////以上内容于 2011-12-17  18:01  添
    加//////////////////////////////////////////
```

1：较“HTML解析-第一版(C/C++)”减少了内存拷贝，速度相对提高很多。

2：代码在VS2008下测试通过。#define _UNICODE #define _WIN32_WINNT 0x0600

3：解析方法：类似于构建一个map表（STL模板库里的map不利于阅读，可以参考MFC类库的CMap），最终组成一个二维的单向链表。

4：CHtmlObject 类负责解析HTML“标记”和“属性”。

[cpp]

```cpp
////////////////////////////////////////////////////////////////////////////////

#pragma once

/*****************************************************************************
created: 2011/12/03
author: hmm7e (hmm7e_z@126.com)

*****************************************************************************/

class CHtmlObject
{
public:
//
static BOOL IsSpace(TCHAR tcLetter);
protected:
struct tagNode
{
LPCTSTR s_pszKey;
LPCTSTR s_pszValue;
struct tagNode * s_pstRight; //attribute of tag
struct tagNode * s_pstNext; //next tag
};
public:
CHtmlObject(void);
virtual ~CHtmlObject(void);
//
enum {CHARSET_UTF8,CHARSET_UNICODE,CHARSET_MULTIBYTE}TextCharset;
protected:
//
tagNode * InnerAllocNode();
void InnerFreeNode(tagNode * lpstNode);
void InnerLinkNextNode(tagNode * lpstNode);
void InnerLinkRightNode(tagNode * lpstTagNode,tagNode * lpstNode);
void InnerCleanupNode();
void InnerCleanupRightNode(tagNode * lpstNode);
public:
//
void AutoTakeSnapshot(PBYTE lpszString,UINT nStringLen);
void TakeSnapshot(PBYTE lpszString,UINT nStringLen,UINT nFromCharset );
void DeleteSnapshot();
//
void Parse();
private:
//
void InnerParse();
LPTSTR InnerSplitComment(tagNode * lpstNode,LPTSTR lpszTagString);
LPTSTR InnerSplitTag(tagNode * lpstNode,LPTSTR lpszTagString);
LPTSTR InnerSplitContent(tagNode * lpstNode,LPTSTR lpszTagString);
LPTSTR InnerSplitText(tagNode * lpstNode,LPTSTR lpszTagString);
LPTSTR InnerSplitScript(tagNode * lpstNode,LPTSTR lpszTagString);
LPTSTR InnerSplitStyle(tagNode * lpstNode,LPTSTR lpszTagString);

protected:
//
LPTSTR m_pszSnapshotBuffer;
UINT m_nSnapshotBufferLen;
UINT m_nSnapshotStringLen;
//
tagNode * m_pstHead;
tagNode * m_pstTail;

};

////////////////////////////////////////////////////////////////////////////////
```

[cpp]

```cpp
////////////////////////////////////////////////////////////////////////////////

#pragma once

/*****************************************************************************
created: 2011/12/03
author: hmm7e (hmm7e_z@126.com)

```

```
09. ********************************************************************************************
10. 
11. 
12. #include "HtmlObject.h"
13. 
14. //
15. BOOL CHtmlObject::IsSpace(TCHAR tcLetter)
16. {
17.     //以下字符在HTML标记里都算是空格。
18.     return (tcLetter == _T(' ') || tcLetter == _T('\r') ||tcLetter == _T('\n') ||tcLetter
19. }
20. 
21. CHtmlObject::CHtmlObject(void)
22. {
23.     m_pszSnapshotBuffer = NULL;
24.     m_nSnapshotBufferLen = 0;
25.     m_nSnapshotStringLen = 0;
26.     m_pstHead = NULL;
27.     m_pstTail = NULL;
28. }
29. 
30. CHtmlObject::~CHtmlObject(void)
31. {
32.     DeleteSnapshot();
33. }
34. //
35. CHtmlObject::tagNode * CHtmlObject::InnerAllocNode()
36. {
37.     CHtmlObject::tagNode * pstResult = new CHtmlObject::tagNode;
38.     if( pstResult )
39.     {
40.         ::ZeroMemory((LPVOID)pstResult,sizeof(CHtmlObject::tagNode));
41.     }
42.     return pstResult;
43. }
44. void CHtmlObject::InnerFreeNode(CHtmlObject::tagNode * lpstNode)
45. {
46.     if( lpstNode )
47.         delete lpstNode;
48. }
49. void CHtmlObject::InnerLinkNextNode(tagNode * lpstNode)
50. {
51.     //链接到“尾”结点。
52.     //1：如果没有“头”节点，那么表示链表是“空”的。
53.     //2：如果已经存“头”节点，那么就链接新节点到“尾”节点，并重新记录“尾”节点指针。
54.     if( m_pstHead == NULL )
55.     {
56.         m_pstHead = lpstNode;
57.         m_pstTail = lpstNode;
58.     }
59.     else
60.     {
61.         m_pstTail->s_pstNext = lpstNode;
62.         m_pstTail = lpstNode;
63.     }
64. 
65. 
66. #ifdef _DEBUG
67. 
68.     if( lpstNode->s_pszKey )
69.     {
70.         ::OutputDebugString(_T("--"));
71.         ::OutputDebugString(lpstNode->s_pszKey);
72.         ::OutputDebugString(_T("--\r\n"));
73.     }
74.     if( lpstNode->s_pszValue )
75.     {
76.         ::OutputDebugString(_T("--"));
77.         ::OutputDebugString(lpstNode->s_pszValue);
78.         ::OutputDebugString(_T("--\r\n"));
79.     }
80. 
81. #endif //_DEBUG
82. 
83. }
84. void CHtmlObject::InnerLinkRightNode(tagNode * lpstTagNode,tagNode * lpstNode)
85. {
86.     //链接到“属性”的“头”节点。
87.     //1：把现有的“属性”链表，链接到当前新节点的下。
```

```
88.     //2：把当前节点做为“头”节点保存。
89.     lpstNode->s_pstRight = lpstTagNode->s_pstRight;
90.     lpstTagNode->s_pstRight = lpstNode;
91.
92. #ifdef _DEBUG
93.     if( lpstNode->s_pszKey )
94.     {
95.         ::OutputDebugString(_T("-->"));
96.         ::OutputDebugString(lpstNode->s_pszKey);
97.         ::OutputDebugString(_T("<--\r\n"));
98.     }
99.     if( lpstNode->s_pszValue )
100.    {
101.        ::OutputDebugString(_T("-->"));
102.        ::OutputDebugString(lpstNode->s_pszValue);
103.        ::OutputDebugString(_T("<--\r\n"));
104.    }
105. #endif //_DEBUG
106. }
107. void CHtmlObject::InnerCleanupNode()
108. {
109.     //循环清除所有节点。如果存在“属性”节点一并清除。
110.     CHtmlObject::tagNode * pstPrev = NULL;
111.     while( m_pstHead )
112.     {
113.         pstPrev = m_pstHead;
114.         m_pstHead = m_pstHead->s_pstNext;
115.         //first
116.         InnerCleanupRightNode(pstPrev);
117.         //second
118.         InnerFreeNode(pstPrev);
119.     }
120.     m_pstHead = NULL;
121.     m_pstTail = NULL;
122. }
123. void CHtmlObject::InnerCleanupRightNode(CHtmlObject::tagNode * lpstNode)
124. {
125.     //循环清除所有“属性”节点。
126.     CHtmlObject::tagNode * pstHead = lpstNode->s_pstRight;
127.     CHtmlObject::tagNode * pstPrev = NULL;
128.     while( pstHead )
129.     {
130.         pstPrev = pstHead;
131.         pstHead = pstHead->s_pstRight;
132.         InnerFreeNode(pstPrev);
133.     }
134.     pstHead = NULL;
135.     pstPrev = NULL;
136. }
137. //
138. void CHtmlObject::AutoTakeSnapshot(PBYTE lpszString,UINT nStringLen)
139. {
140.
141.     if( lpszString && nStringLen > 0)
142.     {
143.         //根据数据头自动判断是否需要转换数据到当前应程所使用的编码。
144.         if( nStringLen >= 2 )
145.         {
146.             if( lpszString[0] == 0xFF && lpszString[1] == 0xFE ) // skip 0xFF,0xFE
147.             {
148.                 TakeSnapshot(lpszString+2,nStringLen-2,CHtmlObject::CHARSET_UNICODE);
149.             }
150.             else if( lpszString[0] == 0xEF && lpszString[1] == 0xBB && lpszString[2] == 0x
151.             {
152.                 TakeSnapshot(lpszString+3,nStringLen-3,CHtmlObject::CHARSET_UTF8);
153.             }
154.             else
155.             {
156.                 TakeSnapshot(lpszString,nStringLen,CHtmlObject::CHARSET_MULTIBYTE);
157.             }
158.         }
159.         else
160.         {
161.             TakeSnapshot(lpszString,nStringLen,CHtmlObject::CHARSET_MULTIBYTE);
162.         }
163.     }
164. }
165. void CHtmlObject::TakeSnapshot(PBYTE lpszString,UINT nStringLen,UINT nFromCharset )
166. {
```

```
    //delete old snapshot
    DeleteSnapshot();

    if( lpszString && nStringLen > 0 )
    {

        //transform to TCHAR

        if( CHtmlHelper::CHARSET_UTF8 == nFromCharset )
        {

#ifdef _UNICODE

            m_nSnapshotBufferLen = nStringLen;
            m_pszSnapshotBuffer = new TCHAR[m_nSnapshotBufferLen];

            ::memset((LPVOID)m_pszSnapshotBuffer,0,m_nSnapshotBufferLen*sizeof(TCHAR));
            m_nSnapshotStringLen = ::MultiByteToWideChar(CP_UTF8,0,
(LPCSTR)lpszString,nStringLen,m_pszSnapshotBuffer,m_nSnapshotBufferLen);
#else
            ::OutputDebugString(_T("no support"));

#endif //_UNICODE

        }
        else if( CHtmlHelper::CHARSET_UNICODE == nFromCharset )
        {

#ifdef _UNICODE

            m_nSnapshotBufferLen = nStringLen;
            m_pszSnapshotBuffer = new TCHAR[m_nSnapshotBufferLen];

            ::memset((LPVOID)m_pszSnapshotBuffer,0,m_nSnapshotBufferLen*sizeof(TCHAR));
            ::memcpy((LPVOID)m_pszSnapshotBuffer,lpszString,nStringLen);

#else

            m_nSnapshotBufferLen = nStringLen/2+1;
            m_pszSnapshotBuffer = new TCHAR[m_nSnapshotBufferLen];

            ::memset((LPVOID)m_pszSnapshotBuffer,0,m_nSnapshotBufferLen*sizeof(TCHAR));
            m_nSnapshotStringLen = ::WideCharToMultiByte(CP_ACP,0,
(LPWSTR)lpszString,nStringLen,
(LPSTR)m_pszSnapshotBuffer,m_nSnapshotBufferLen,NULL,NULL);

#endif //_UNICODE

        }
        else
        {

#ifdef _UNICODE

            m_nSnapshotBufferLen = nStringLen;
            m_pszSnapshotBuffer = new TCHAR[m_nSnapshotBufferLen];

            ::memset(m_pszSnapshotBuffer,0,m_nSnapshotBufferLen*sizeof(TCHAR));
            m_nSnapshotStringLen = ::MultiByteToWideChar(CP_ACP,0,
(LPCSTR)lpszString,nStringLen,m_pszSnapshotBuffer,m_nSnapshotBufferLen);
#else

            m_nSnapshotBufferLen = nStringLen;
            m_pszSnapshotBuffer = new TCHAR[m_nSnapshotBufferLen];

            ::memset((LPVOID)m_pszSnapshotBuffer,0,m_nSnapshotBufferLen*sizeof(TCHAR));
            ::memcpy((LPVOID)m_pszSnapshotBuffer,lpszString,nStringLen);

#endif //_UNICODE

        }
    }
}
void CHtmlObject::DeleteSnapshot()
{
    //先清除树型表。
    InnerCleanupNode();

    if( m_pszSnapshotBuffer )
```

```
242.         delete []m_pszSnapshotBuffer;
243.
244.     m_pszSnapshotBuffer = NULL;
245.     m_nSnapshotBufferLen = 0;
246.     m_nSnapshotStringLen = 0;
247. }
248. //
249. void CHtmlObject::Parse()
250. {
251. #ifdef _AFX
252.     CString strTrace;
253.     strTrace.Format(_T("CHtmlObject::Parse() --begin-->(%d)\r\n"),::GetTickCount());
254.     ::OutputDebugString(strTrace);
255. #endif //_AFX
256.
257.     InnerParse();
258.
259. #ifdef _AFX
260.     strTrace.Format(_T("CHtmlObject::Parse() --end-->(%d)\r\n"),::GetTickCount());
261.     ::OutputDebugString(strTrace);
262. #endif //_AFX
263. }
264. //
265. void CHtmlObject::InnerParse()
266. {
267.     LPTSTR pszFind = m_pszSnapshotBuffer;
268.
269.     //跳过所有“空格”
270.     while( *pszFind != _T('\0') && CHtmlObject::IsSpace(*pszFind) )
271.     {
272.         //下一个字符
273.         pszFind++;
274.     }
275.     //直到碰到'\0'就退出
276.     do
277.     {
278.         // 不是“\0”，并且第一个字符为“<”则置换为“\0”，否则什么也不做。
279.         //这么写的原因就在于InnerSplitContent()返回后 “<”可能已经被置换成“\0”。
280.         if( *pszFind != _T('\0') && *pszFind == _T('<') )
281.         {
282.             //把“<”置换为“\0”,做为结尾。
283.             *pszFind = _T('\0');
284.             //下一个字符。
285.             pszFind++;
286.         }
287.
288.         // 不是“\0”
289.         if( *pszFind != _T('\0') )
290.         {
291.             //是否为注释
292.             if( *pszFind == _T('!') )
293.             {
294.                 //申请一个点节。
295.                 tagNode *pstNode = InnerAllocNode();
296.                 //解析注释，返回的是注释后面的内容。
297.                 pszFind = InnerSplitComment(pstNode,pszFind);
298.                 //链接到“链表”。（下）
299.                 InnerLinkNextNode(pstNode);
300.             }
301.             else
302.             {
303.                 //申请一个点节。
304.                 tagNode *pstNode = InnerAllocNode();
305.                 //解析tag，返回的是tag后面的内容。
306.                 pszFind = InnerSplitTag(pstNode,pszFind);
307.                 //解析content返回的是content后面的内容。
308.                 pszFind = InnerSplitContent(pstNode,pszFind);
309.                 //链接到“链表”。（下）
310.                 InnerLinkNextNode(pstNode);
311.             }
312.         }
313.
314.     }while( *pszFind!= _T('\0') );
315. }
316. LPTSTR CHtmlObject::InnerSplitComment(CHtmlObject::tagNode * lpstNode,LPTSTR lpszTagString
317. {
318.     LPTSTR pszFind = lpszTagString;
319.     //指向注释开头(已经跳过“<”字符)
320.     lpstNode->s_pszKey = pszFind;
```

```
321.     //如果为 <!-- *** -->
322.     if( ::_tcsnicmp(pszFind+1,_T("--"),2) == 0 )
323.     {
324.         //跳过注释标记“头”，开始查找。
325.         pszFind += 3;
326.         //查找到注释结尾，并给结尾加“\0”。
327.         while( ::_tcsnicmp(pszFind,_T("-->"),3) != 0 )
328.         {
329.             //下一个字符
330.             pszFind++;
331.         }
332.         //不是“\0”
333.         if( *pszFind != _T('\0') )
334.         {
335.             //把“>”置换为“\0”,做为注释结尾
336.             *(pszFind+2) = _T('\0');
337.             //指向新的节点或内容。
338.             pszFind += 3;
339.         }
340.
341.     }
342.     //否则为 <! *** >
343.     else
344.     {
345.         //查找到注释结尾，并给结尾加“\0”。
346.         while( *pszFind != _T('\0') && *pszFind != _T('>') )
347.         {
348.             //下一个字符
349.             pszFind++;
350.         }
351.         //不能是“\0”
352.         if( *pszFind != _T('\0') )
353.         {
354.             //把“>”置换为“\0”,做为注释结尾。
355.             *pszFind = _T('\0');
356.             //指向新的节点或内容。
357.             pszFind++;
358.         }
359.     }
360.
361.     //找到一个“<”
362.     while( *pszFind != _T('\0') && *pszFind != _T('<') )
363.     {
364.         //下一个字符
365.         pszFind++;
366.     }
367.
368.     return pszFind;
369. }
370. LPTSTR CHtmlObject::InnerSplitTag(CHtmlObject::tagNode * lpstNode,LPTSTR lpszTagString)
371. {
372.     LPTSTR pszFind = lpszTagString;
373.
374.     //指向开头(已经跳过“<”字符)
375.     lpstNode->s_pszKey = pszFind;
376.     //查找tag结尾，并给结尾加“\0”。
377.     while( *pszFind != _T('\0') && *pszFind != _T('>') && !CHtmlObject::IsSpace(*pszFind)
378.     {
379.         //下一个字符
380.         pszFind++;
381.     }
382.
383.     //不是“\0”
384.     if( *pszFind != _T('\0') )
385.     {
386.         if( *pszFind == _T('>') )
387.         {
388.             //把“>”置换为“\0”,做为注释结尾。
389.             *pszFind = _T('\0');
390.             //指向新的节点或内容。
391.             pszFind++;
392.             //此tag没有属性,什么也不做了。
393.
394.         }
395.         else
396.         {
397.             //把“space,\r,\n,\t ”置换为“\0”,做为注释结尾。
398.             *pszFind = _T('\0');
399.             //指向新的节点或内容。
```

```
400.            pszFind++;
401.
402.            //如果不是结束标记，表示此tag有“属性”还需要解析“属性”。
403.            if( *lpstNode->s_pszKey != _T('/') )
404.            {
405.                //跳过所有“空格”,找到第一个属性。
406.                while( *pszFind != _T('\0') && CHtmlObject::IsSpace(*pszFind) )
407.                {
408.                    //下一个字符
409.                    pszFind++;
410.                }
411.                //循环分析“属性”。
412.                while( *pszFind != _T('\0') && *pszFind != _T('<') && *pszFind != _T('>')
413.                {
414.                    //例：
415.                    // key="value" key=value
416.                    //跳过空格
417.                    while( *pszFind != _T('\0') && CHtmlObject::IsSpace(*pszFind) )
418.                    {
419.                        //下一个字符
420.                        pszFind++;
421.                    }
422.
423.                    //不是“\0”
424.                    if( *pszFind != _T('\0') )
425.                    {
426.                        //申请一个点节。
427.                        tagNode *pstAttributeNode = InnerAllocNode();
428.                        //指向“属性”Key。
429.                        pstAttributeNode->s_pszKey = pszFind;
430.
431.                        //查找key的末尾.
432.                        while( *pszFind != _T('\0') && *pszFind != _T('=') && *pszFind !=
433.                        {
434.                            //下一个字符
435.                            pszFind++;
436.                        }
437.                        //不是“\0”
438.                        if( *pszFind != _T('\0') )
439.                        {
440.                            if( *pszFind == _T('>') )
441.                            {
442.                                //把“>”置换为“\0”,做为结尾。
443.                                *pszFind = _T('\0');
444.                                //指向新的节点或内容。
445.                                pszFind++;
446.                                //链接到“链表”（右）。
447.                                InnerLinkRightNode(lpstNode,pstAttributeNode);
448.                                //已经碰到“>”,需要跳出。
449.                                break;
450.                            }
451.                            else
452.                            {
453.                                //把“=”置换为“\0”,做为结尾。
454.                                *pszFind = _T('\0');
455.                                //指向新的节点或内容。
456.                                pszFind++;
457.
458.                                //不是“\0”
459.                                if( *pszFind != _T('\0') )
460.                                {
461.                                    if( *pszFind == _T('"') )
462.                                    {
463.                                        //跳过“"”
464.                                        pszFind++;
465.                                        //指向“属性”key的Value。
466.                                        pstAttributeNode->s_pszValue = pszFind;
467.
468.                                        //查找Value的末尾.
469.                                        while( *pszFind != _T('\0') && *pszFind != _T('\"'
470.                                        {
471.                                            //下一个字符
472.                                            pszFind++;
473.                                        }
474.                                        //不是“\0”
475.                                        if( *pszFind != _T('\0') )
476.                                        {
477.                                            //把“",> ”置换为“\0”,做为结尾。
478.                                            *pszFind = _T('\0');
```

```
                                        //指向新的节点或内容。
                                        pszFind++;
                                    }
                                }
                                else if( *pszFind == _T('\'') )
                                {
                                    //跳过“'”
                                    pszFind++;
                                    //指向“属性”key的Value。
                                    pstAttributeNode->s_pszValue = pszFind;

                                    //查找Value的末尾.
                                    while( *pszFind != _T('\0') && *pszFind != _T('\''
                                    {
                                        //下一个字符
                                        pszFind++;
                                    }
                                    //不是“\0”
                                    if( *pszFind != _T('\0') )
                                    {
                                        //把“",<space> ”置换为“\0”,做为结尾。
                                        *pszFind = _T('\0');
                                        //指向新的节点或内容。
                                        pszFind++;
                                    }
                                }
                                else
                                {
                                    //指向“属性”key的Value。
                                    pstAttributeNode->s_pszValue = pszFind;
                                    //查找Value的末尾.
                                    while( *pszFind != _T('\0') && *pszFind != _T(' ')
                                    {
                                        //下一个字符
                                        pszFind++;
                                    }
                                    //不是“\0”
                                    if( *pszFind != _T('\0') )
                                    {
                                        //把“ ”置换为“\0”,做为结尾。
                                        *pszFind = _T('\0');
                                        //指向新的节点或内容。
                                        pszFind++;
                                    }
                                }

                            }
                        }

                    }

                    //链接到“链表”（右）。
                    InnerLinkRightNode(lpstNode,pstAttributeNode);
                }
            }

            //跳过这个无用的字符。
            if( *pszFind == _T('>') )
            {
                //指向新的节点或内容。
                pszFind++;
            }
        }
    }

    return pszFind;
}
LPTSTR CHtmlObject::InnerSplitContent(CHtmlObject::tagNode * lpstNode,LPTSTR lpszTagString
{
    LPTSTR pszFind = lpszTagString;

    if( ::_tcsnicmp(lpstNode->s_pszKey,_T("script"),6) == 0 )
    {
        pszFind = InnerSplitScript(lpstNode,pszFind);
    }
    else if( ::_tcsnicmp(lpstNode->s_pszKey,_T("style"),5) == 0 )
    {
        pszFind = InnerSplitStyle(lpstNode,pszFind);
```

```
    }
    else
    {
        pszFind = InnerSplitText(lpstNode,pszFind);
    }

    return pszFind;
}
LPTSTR CHtmlObject::InnerSplitText(CHtmlObject::tagNode * lpstNode,LPTSTR lpszTagString)
{
    LPTSTR pszFind = lpszTagString;

    //跳过所有“空格”
    while( *pszFind != _T('\0') && CHtmlObject::IsSpace(*pszFind) )
    {
        //下一个字符
        pszFind++;
    }

    //如果 _T('<') 表示没有文本。
    if( *pszFind != _T('<') )
    {
        //指向可见文本。
        lpstNode->s_pszValue = pszFind;
        //查找文本结尾。
        while( *pszFind != _T('\0') && *pszFind != _T('<') && !CHtmlObject::IsSpace(*pszFi
        {
            //下一个字符
            pszFind++;
        }
        //不是“\0”
        if( *pszFind != _T('\0') )
        {
            if( *pszFind == _T('<') )
            {
                //把“<”置换为“\0”,做为结尾。
                *pszFind = _T('\0');
                //指向新的节点或内容。
                pszFind++;
            }
            else
            {
                //把“space,\r,\n,\t,”置换为“\0”,做为结尾。
                *pszFind = _T('\0');
                //指向新的节点或内容。
                pszFind++;

                //找到一个“<”
                while( *pszFind != _T('\0') && *pszFind != _T('<') )
                {
                    //下一个字符
                    pszFind++;
                }
            }
        }
    }

    return pszFind;
}
LPTSTR CHtmlObject::InnerSplitScript(tagNode * lpstNode,LPTSTR lpszTagString)
{
    LPTSTR pszFind = lpszTagString;


#define  SCRIPT_MARK_MAX            1024
    UINT nMarkIndex = 0;
    TCHAR szMark[SCRIPT_MARK_MAX] = {_T('\0')}; //max 1024

    //跳过所有“空格”
    while( *pszFind != _T('\0') && CHtmlObject::IsSpace(*pszFind) )
    {
        //下一个字符
        pszFind++;
    }

    if( *pszFind != _T('\0') && *pszFind != _T('<') )
    {
        //指向可见文本。
        lpstNode->s_pszValue = pszFind;
```

```
637.
638.        while( *pszFind != _T('\0') )
639.        {
640.            //如果字符被“'，""”包围则为字符串，这期间不计算注释。
641.            if( szMark[nMarkIndex] != _T('\'') && szMark[nMarkIndex] != _T('\"') )
642.            {
643.                //如果是// abc 则跳过。
644.                if( ::_tcsnicmp(pszFind,_T("//"),2) == 0   )
645.                {
646.                    //跳过注释“头”。
647.                    pszFind +=2;
648.                    //查找注释“尾”。
649.                    while( *pszFind != _T('\0') && *pszFind != _T('\n') )
650.                    {
651.                        pszFind++;
652.                    }
653.                    //跳过注释“尾”。
654.                    if( *pszFind != _T('\0')  )
655.                        pszFind++;
656.
657.                }
658.                //如果是/* abc */则跳过。
659.                else if( ::_tcsnicmp(pszFind,_T("/*"),2) == 0   )
660.                {
661.                    //跳过注释“头”。
662.                    pszFind +=2;
663.                    //查找注释“尾”。
664.                    while( ::_tcsnicmp(pszFind,_T("*/"),2) != 0 )
665.                    {
666.                        pszFind++;
667.                    }
668.                    //跳过注释“尾”。
669.                    if( *pszFind != _T('\0')  )
670.                        pszFind +=2;
671.                }
672.
673.            }
674.
675.            if( *pszFind == _T('\\') &&
676.                ( *(pszFind+1) == _T('\\') ||
677.                *(pszFind+1) == _T('(') || *(pszFind+1) == _T(')') ||
678.                *(pszFind+1) == _T('[') || *(pszFind+1) == _T(']') ||
679.                *(pszFind+1) == _T('{') || *(pszFind+1) == _T('}') ||
680.                *(pszFind+1) == _T('\'') ||
681.                *(pszFind+1) == _T('\"') ) )
682.            {
683.                //转意字符
684.                pszFind+=2;
685.
686.            }
687.            else if( *pszFind == _T('{') || *pszFind == _T('(') || *pszFind == _T('[') ||
688.            {
689.                if( szMark[nMarkIndex] != _T('\'') && szMark[nMarkIndex] != _T('\"') )
690.                {
691.                    if( nMarkIndex < SCRIPT_MARK_MAX )
692.                    {
693.                        if( nMarkIndex == 0 && szMark[nMarkIndex] == _T('\0')  )
694.                            szMark[nMarkIndex] = *pszFind;
695.                        else
696.                            szMark[++nMarkIndex] = *pszFind;
697.                    }
698.                }
699.                else if( szMark[nMarkIndex] == *pszFind )
700.                {
701.                    if( nMarkIndex >0  )
702.                        szMark[nMarkIndex--] = _T('\0');
703.                    else
704.                        szMark[nMarkIndex] = _T('\0');
705.                }
706.                pszFind++;
707.
708.            }
709.            else if( *pszFind == _T('}') )
710.            {
711.                if( szMark[nMarkIndex] == _T('{') )
712.                {
713.                    if( nMarkIndex >0  )
714.                        szMark[nMarkIndex--] = _T('\0');
715.                    else
```

```
716.                        szMark[nMarkIndex] = _T('\0');
717.                    }
718.                    pszFind++;
719.                }
720.                else if( *pszFind == _T(')') )
721.                {
722.                    if( szMark[nMarkIndex] == _T('(') )
723.                    {
724.                        if( nMarkIndex >0  )
725.                            szMark[nMarkIndex--] = _T('\0');
726.                        else
727.                            szMark[nMarkIndex] = _T('\0');
728.                    }
729.                    pszFind++;
730.                }
731.                else if( *pszFind == _T(']') )
732.                {
733.                    if( szMark[nMarkIndex] == _T('[') )
734.                    {
735.                        if( nMarkIndex >0  )
736.                            szMark[nMarkIndex--] = _T('\0');
737.                        else
738.                            szMark[nMarkIndex] = _T('\0');
739.                    }
740.                    pszFind++;
741.                }
742.                else if( *pszFind == _T('<') && szMark[0] == _T('\0') )  //nMarkIndex == 0 &&
743.                {
744.                    //把“<”置换为“\0”,做为结尾。
745.                    *pszFind = _T('\0');
746.                    //指向新的节点或内容。
747.                    pszFind++;
748.                    break;
749.                }
750.                else
751.                {
752.                    pszFind++;
753.                }
754.            }
755.        }
756.
757.        return pszFind;
758.    }
759.    LPTSTR CHtmlObject::InnerSplitStyle(CHtmlObject::tagNode * lpstNode,LPTSTR lpszTagString)
760.    {
761.        LPTSTR pszFind = lpszTagString;
762.
763.    #define  STYLE_MARK_MAX         1024
764.        UINT nMarkIndex = 0;
765.        TCHAR szMark[STYLE_MARK_MAX] = {_T('\0')}; //max 1024
766.
767.        //跳过所有“空格”
768.        while( *pszFind != _T('\0') && CHtmlObject::IsSpace(*pszFind) )
769.        {
770.            //下一个字符
771.            pszFind++;
772.        }
773.
774.        if( *pszFind != _T('\0') && *pszFind != _T('<') )
775.        {
776.            //指向可见文本。
777.            lpstNode->s_pszValue = pszFind;
778.
779.            while( *pszFind != _T('\0') )
780.            {
781.                //如果字符被“(，'，"”包围则为字符串，这期间不计算注释。
782.                if( szMark[nMarkIndex] != _T('(') && szMark[nMarkIndex] != _T('\'') && szMark[
783.                {
784.                    //如果是/* abc */则跳过。
785.                    if( ::_tcsnicmp(pszFind,_T("/*"),2) == 0   )
786.                    {
787.                        //跳过注释“头”，查找注释“尾”。
788.                        pszFind +=2;
789.                        while( ::_tcsnicmp(pszFind,_T("*/"),2) != 0 )
790.                        {
791.                            pszFind++;
792.                        }
793.                        //跳过注释“尾”。
794.                        if( *pszFind != _T('\0')  )
```

```
                        pszFind +=2;
                    }
                }


                if( *pszFind == _T('{') || *pszFind == _T('(') || *pszFind == _T('[') ||  (*ps
                {
                    if( szMark[nMarkIndex] != _T('\'') && szMark[nMarkIndex] != _T('\"') )
                    {
                        if( nMarkIndex < STYLE_MARK_MAX )
                        {
                            if( nMarkIndex == 0 && szMark[nMarkIndex] == _T('\0')  )
                                szMark[nMarkIndex] = *pszFind;
                            else
                                szMark[++nMarkIndex] = *pszFind;
                        }
                    }
                    else if( szMark[nMarkIndex] == *pszFind )
                    {
                        if( nMarkIndex >0  )
                            szMark[nMarkIndex--] = _T('\0');
                        else
                            szMark[nMarkIndex] = _T('\0');
                    }
                    pszFind++;

                }
                else if( *pszFind == _T('}') )
                {
                    if( szMark[nMarkIndex] == _T('{') )
                    {
                        if( nMarkIndex >0  )
                            szMark[nMarkIndex--] = _T('\0');
                        else
                            szMark[nMarkIndex] = _T('\0');
                    }
                    pszFind++;
                }
                else if( *pszFind == _T(')') )
                {
                    if( szMark[nMarkIndex] == _T('(') )
                    {
                        if( nMarkIndex >0  )
                            szMark[nMarkIndex--] = _T('\0');
                        else
                            szMark[nMarkIndex] = _T('\0');
                    }
                    pszFind++;
                }
                else if( *pszFind == _T(']') )
                {
                    if( szMark[nMarkIndex] == _T('[') )
                    {
                        if( nMarkIndex >0  )
                            szMark[nMarkIndex--] = _T('\0');
                        else
                            szMark[nMarkIndex] = _T('\0');
                    }
                    pszFind++;
                }
                else if( *pszFind == _T('<') && szMark[0] == _T('\0') )  //nMarkIndex == 0 &&
                {
                    //把“<”置换为“\0”,做为结尾。
                    *pszFind = _T('\0');
                    //指向新的节点或内容。
                    pszFind++;
                    break;
                }
                else
                {
                    pszFind++;
                }
            }

        }

        return pszFind;
    }

```

```
874. ////////////////////////////////////////////////////////////////////////////////////////
```

5：CHtmlHelper做为派生类，负责读取解析后的“标记”和“属性”。这里只写了两个方法，实际应用请自行添加。

```cpp
////////////////////////////////////////////////////////////////////////////////////////

#pragma once

/*****************************************************************************************
created: 2011/12/03
author: hmm7e (hmm7e_z@126.com)

*****************************************************************************************/

#include "HtmlObject.h"

class CHtmlHelper:public CHtmlObject
{
public:
    CHtmlHelper(void);
    virtual ~CHtmlHelper(void);
public:
    //
    LPCTSTR GetFirstLink();
    LPCTSTR GetNextLink();
    LPCTSTR GetFirstContent();
    LPCTSTR GetNextContent();
    LPCTSTR SearchText(LPCTSTR lpszText);
protected:
    //
    CHtmlObject::tagNode * m_pstCur;
};



////////////////////////////////////////////////////////////////////////////////////////
```

```cpp
////////////////////////////////////////////////////////////////////////////////////////

#pragma once

/*****************************************************************************************
created: 2011/12/03
author: hmm7e (hmm7e_z@126.com)

*****************************************************************************************/


#include "HtmlHelper.h"

#pragma warning(disable: 4996)

CHtmlHelper::CHtmlHelper()
{

}
CHtmlHelper::~CHtmlHelper()
{

}
//
LPCTSTR CHtmlHelper::GetFirstLink()
{
    LPCTSTR pszResult = NULL;

    m_pstCur = m_pstHead;

    while( m_pstCur && !pszResult )
    {
        if( 0 != ::_tcsnicmp(m_pstCur->s_pszKey,_T("script"),6) &&
            0 != ::_tcsnicmp(m_pstCur->s_pszKey,_T("style"),5) )
        {
            CHtmlObject::tagNode * pstAttributeCur = m_pstCur->s_pstRight;
```

```
            while( pstAttributeCur )
            {
                if( 0 == ::_tcsnicmp(pstAttributeCur->s_pszKey,_T("href"),4) ||
                    0 == ::_tcsnicmp(pstAttributeCur->s_pszKey,_T("src"),3) )
                {
                    //return
                    pszResult = pstAttributeCur->s_pszValue;
                    break ;
                }
                else
                {
                    pstAttributeCur = pstAttributeCur->s_pstRight;
                }
            }
        }
        m_pstCur = m_pstCur->s_pstNext;
    }

    return pszResult;
}
LPCTSTR CHtmlHelper::GetNextLink()
{
    LPCTSTR pszResult = NULL;

    while( m_pstCur && !pszResult )
    {
        if( 0 != ::_tcsnicmp(m_pstCur->s_pszKey,_T("script"),6) &&
            0 != ::_tcsnicmp(m_pstCur->s_pszKey,_T("style"),5) )
        {
            CHtmlObject::tagNode * pstAttributeCur = m_pstCur->s_pstRight;
            while( pstAttributeCur )
            {
                if( 0 == ::_tcsnicmp(pstAttributeCur->s_pszKey,_T("href"),4) ||
                    0 == ::_tcsnicmp(pstAttributeCur->s_pszKey,_T("src"),3) )
                {
                    //return
                    pszResult = pstAttributeCur->s_pszValue;
                    break ;
                }
                else
                {
                    pstAttributeCur = pstAttributeCur->s_pstRight;
                }
            }
        }

        m_pstCur = m_pstCur->s_pstNext;
    }

    return pszResult;
}
LPCTSTR CHtmlHelper::GetFirstContent()
{
    LPCTSTR pszResult = NULL;

    m_pstCur = m_pstHead;

    while( m_pstCur && !pszResult )
    {
        if( 0 != ::_tcsnicmp(m_pstCur->s_pszKey,_T("script"),6) &&
            0 != ::_tcsnicmp(m_pstCur->s_pszKey,_T("style"),5) )
        {
            if( m_pstCur->s_pszValue )
                pszResult = m_pstCur->s_pszValue;
        }

        m_pstCur = m_pstCur->s_pstNext;
    }

    return pszResult;
}
LPCTSTR CHtmlHelper::GetNextContent()
{
    LPCTSTR pszResult = NULL;

    while( m_pstCur && !pszResult )
    {
        if( 0 != ::_tcsnicmp(m_pstCur->s_pszKey,_T("script"),6) &&
            0 != ::_tcsnicmp(m_pstCur->s_pszKey,_T("style"),5) )
```

```
116.         {
117.             if( m_pstCur->s_pszValue )
118.                 pszResult = m_pstCur->s_pszValue;
119.         }
120.
121.         m_pstCur = m_pstCur->s_pstNext;
122.     }
123.
124.     return pszResult;
125. }
126. //
127. LPCTSTR CHtmlHelper::SearchText(LPCTSTR lpszText)
128. {
129.     LPCTSTR pszResult = NULL;
130.
131.     CHtmlObject::tagNode *pstCur = m_pstHead;
132.
133.     while( pstCur && !pszResult)
134.     {
135.         if( 0 != ::_tcsnicmp(pstCur->s_pszKey,_T("script"),6) &&
136.             0 != ::_tcsnicmp(pstCur->s_pszKey,_T("style"),5) )
137.         {
138.             if( pstCur->s_pszValue )
139.             {
140.                 if( (NULL != ::StrStrI(pstCur->s_pszValue,lpszText)) )
141.                     pszResult = pstCur->s_pszValue;
142.             }
143.         }
144.
145.         pstCur = pstCur->s_pstNext;
146.     }
147.
148.     return pszResult;
149. }
150.
151.
152.
153. #pragma warning(default: 4996)
154.
155.
156.
157. ////////////////////////////////////////////////////////////////////////////////
```

分享到：

查看评论

9楼 li_mountain 2011-12-28 19:36发表

mark一下，有空仔细看看！^_^

8楼 feimashenhua 2011-12-27 18:04发表

谢谢lz分享！

7楼 ha8211 2011-12-26 15:22发表

谢谢分享！

6楼 pet 2011-12-22 21:12发表

我先学习下代码，呵呵，然后再发表评论

5楼 love_qq 2011-12-20 12:21发表

有关于HTML解析方面的书籍吗

4楼 Donneyming 2011-12-19 12:09发表

我看下吧 争取弄点东西出来 楼主还是很厉害的

3楼 lance 2011-12-18 16:28发表

楼主该去看看自动机相关理论，再写这程序。

2楼 ljw852582663 2011-12-17 14:54发表

谢谢分享！！！

1楼 coding_or_coded 2011-12-17 11:02发表

这种文章，首先应该给读者一个你设计的路线，就比如说你做这个东西的时候的思考过程，不然直接贴代码，很少有人愿意静下心来看的，不巧，我静下来看了，还不错的文章

Re: HMM7E 2011-12-17 16:23发表

回复coding_or_coded：多谢。

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

更多招聘职位