

个人资料




liigo

访问：398425次
积分：7376分
排名：第287名

原创：235篇 转载：10篇
译文：0篇 评论：1223条

Google plus + Liigo

 +Liigo Zhuang

文章搜索

阅读排行

- Microsoft Visual C++... (21989)
- 基本的HTML文本解析器的设计和实现（C... (16362)
- “易语言.飞扬”十分钟入门教程 (15886)
- 易语言“非主流”，杀毒软件“躲猫猫” (9179)
- 遇到C语言相关的两个问题让我心情压抑 (8385)
- 多家权威机构、几十篇权威证据证明：Jav... (8058)
- 我所期待的易语言2007 (7941)
- GDB十分钟教程 (6937)
- liigo：2010年底平板电脑（MID... (6096)
- [原创] Commons-logging... (6017)

推荐文章

精创之作《雷神的微软平台安全宝典》诚邀译者

移动业界领袖会议·上海·6.20

CSDN博客频道“移动开发之我见”主题征文活动

【分享季1】：网友推荐130个经典资源，分享再赠分！

基本的HTML文本解析器的设计和实现（C/C++源码），图文并茂

分类：重复发明轮子 C/C++ 源代码 liigo Parser

2011-01-19 23:28

16363人阅读

评论(62)

收藏

举报

作者：庄晓立 (liigo)

日期：2011-1-19

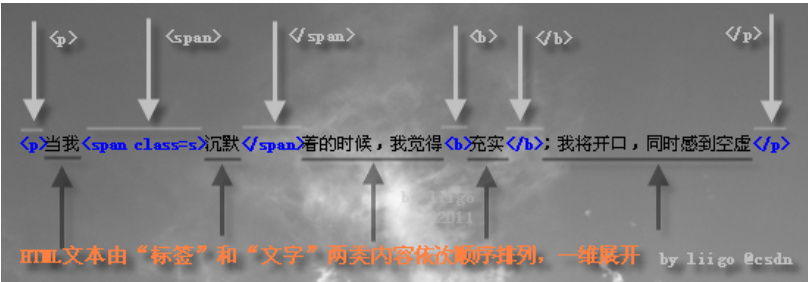
原创链接：<http://blog.csdn.net/liigo/archive/2011/01/19/6153829.aspx>

转载请保持本文完整性，并注明出处：<http://blog.csdn.net/liigo>

关键字：HTML，解析器(Parser)，节点(Node)，标签(Tag)

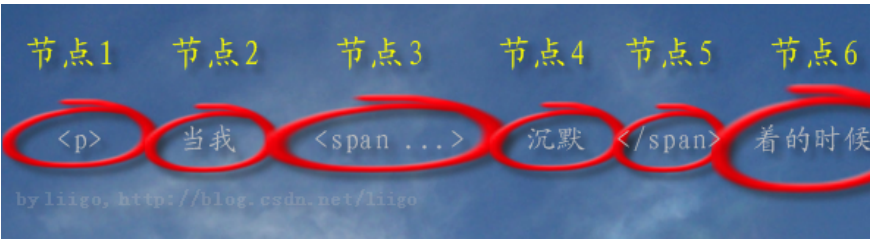
这是进入2011年以来，本人(liigo)“重复发明轮子”系列博文中的最新一篇。本文主要探讨如何设计和实现一个基本的HTML文本解析器。

众所周知，HTML是结构化文档(Structured Document)，由诸多标签（<p>等）嵌套形成的著名的文档对象模型（DOM, Document Object Model），是显而易见的树形多层次结构。如果带着这种思路看待HTML、编写HTML解析器，无疑将导致问题复杂化。不妨从另一视角俯视HTML文本，视其为二维线状结构：诸多单一节点的顺序排列。仔细审视任何一段HTML文本，以左右尖括号（<和>）为边界，会发现HTML文本被天然地分割为：一个标签（Tag），接一段普通文字，再一个标签，再一段普通文字..... 如下图所示：



标签有两种，开始标签（如<p>）和结束标签（</p>），它们和普通文字一起，顺序排列，共同构成了HTML文本的全部。

为了再次简化编程模型，我(liigo)继续将“开始标签”“结束标签”“普通文字”三者统一抽象归纳为“节点”（HtmlNode），相应的，“节点”有三种类型，要么是开始标签，要么是结束标签，要么是普通文字。现在，HTML在我们眼里更加单纯了，它就是“节点”的线性顺序组合，是一维的“节点”数组。如下图所示：HTML文本 = 节点1 + 节点2 + 节点3 +



在正式编码之前，先确定好“节点”的数据结构。作为“普通文字”节点，需要记录一个文本(text)；作为“标签”节点，需要记录标签名称(tagName)、标签类型(tagType)、所有属性值(props)；另外还要有个类型(type)以便区分该节点

*云存储的故事——元数据据

最新评论

在 IIS 5.1 中配置使用 ISAPI 扩展程序, XP sp2, [多图]
liigo: 我 (Liigo) 今天安装IIS后遇到错误: Server Application Error The ...
升级易语言支持库保证向下兼容性的几点总结
liigo: @glbosom: 我文中还特别强调了对于已经编译好的EXE的兼容性 (兼容点4、5)。你这办法显然是...
亲爱的小六E60啊, 你还好吗? 可怜的小六
liigo: @bywayboy: 是啊, 怀念那个时候
升级易语言支持库保证向下兼容性的几点总结
glbosom: 其实易语言做全面的改版对于兼容性的问题,可以像c++一样当判断是旧版本的话直接提示用户转化成一个新版...
有朋友说liigo的名片不错, 拿出来晒晒, 看图
liurendabing: 简单大方 有意思 有意境
有朋友说liigo的名片不错, 拿出来晒晒, 看图
liigo: 顶。名片作者姜岩, 同事。
升级易语言支持库保证向下兼容性的几点总结
liigo: 刚刚把第五段补完了: 五, 为支持库增加常量, 或为枚举类型增加常量成员
修改Go语言(golang)编译器源代码让它支持UTF-8 BOM
liigo: @bywayboy:HTML文件的编码可以写到meta标签里面, 而且这是标准做法。但编程语言的源代码...
基本的HTML文本解析器的设计和实现 (C/C++源码), 图文并茂
liigo: @litongli1:参见文章末尾的补记
亲爱的小六E60啊, 你还好吗? 可怜的小六
bywayboy: 哈哈..... 想当年, 傲气雄鹰.....

评论排行

易语言“非主流”, 杀毒软件“躲猫猫” (116)
基本的HTML文本解析器的设计和实现 (C... (62)
简析移动领域内微软(Microsoft)... (55)
多家权威机构、几十篇权威证据证明: Jav... (55)
遇到C语言相关的两个问题让我心情压抑 (47)
liigo: 2010年底平板电脑 (MID... (34)
Microsoft Visual C++... (22)
2010春节期间易语言论坛大事记, 5.0... (21)
用易语言随手编写闹钟程序, 轻松解决扣奖金... (20)

点是普通文字、开始标签还是结束标签。这其中固然有些冗余信息, 比如对标签来说不需要记录文本, 对普通文字来说又不需要记录标签名称、属性值等, 不过无伤大雅, 简洁的编程模型是最大的诱惑。用C/C++语言语法表示如下:

[cpp]

```
01. enum HtmlNodeType
02. {
03.     NODE_UNKNOWN = 0,
04.     NODE_START_TAG,
05.     NODE_CLOSE_TAG,
06.     NODE_CONTENT,
07. };
08. enum HtmlTagType
09. {
10.     TAG_UNKNOWN = 0,
11.     TAG_A, TAG_DIV, TAG_FONT, TAG_IMG, TAG_P, TAG_SPAN, TAG_BR, TAG_B, TAG_I, TAG_HR,
12. };
13. struct HtmlNodeProp
14. {
15.     WCHAR* szName;
16.     WCHAR* szValue;
17. };
18. #define MAX_HTML_TAG_LENGTH (15)
19. struct HtmlNode
20. {
21.     HtmlNodeType type;
22.     HtmlTagType tagType;
23.     WCHAR tagName[MAX_HTML_TAG_LENGTH+1];
24.     WCHAR* text;
25.     int propCount;
26.     HtmlNodeProp* props;
27. };
```

具体到编写程序代码, 要比想象中容易的多。编码的核心要点是, 以左右尖括号 (<和>) 为边界自然分割标签和普通文字。左右尖括号之间的当然是标签节点 (开始标签或结束标签), 左尖括号(<)之前 (直到前一个右尖括号或开头)、右尖括号(>)之后 (直到后一个左尖括号或结尾) 的显然是普通文字节点。区分开始标签或结束标签的关键点是, 看左尖括号(<)后面第一个非空白字符是否为'/'。对于开始标签, 在标签名称后面, 间隔至少一个空白字符, 可能会有形式为"key1=value1 key2=value2 key3"的属性表, 关于属性表, 后文有专门的函数负责解析。此外有一点要注意, 属性值一般有引号括住, 引号内出现的左右尖括号应该不被视为边界分隔符。

下面就是负责把HTML文本解析为一个个节点 (HtmlNode) 的核心代码 (不足百行, 够精简吧):

[cpp]

```
01. void HtmlParser::ParseHtml(const WCHAR* szHtml)
02. {
03.     m_html = szHtml ? szHtml : L"";
04.     freeHtmlNodes();
05.     if(szHtml == NULL || *szHtml == L'/' || *szHtml == L'\0') return;
06.     WCHAR* p = (WCHAR*) szHtml;
07.     WCHAR* s = (WCHAR*) szHtml;
08.     HtmlNode* pNode = NULL;
09.     WCHAR c;
10.     bool bInQuotes = false;
11.     while( c = *p )
12.     {
13.         if(c == L'/' )
14.         {
15.             bInQuotes = !bInQuotes;
16.             p++; continue;
17.         }
18.         if(bInQuotes)
19.         {
20.             p++; continue;
21.         }
22.         if(c == L'<')
23.         {
24.             if(p > s)
25.             {
26.                 //Add Text Node
27.                 pNode = NewHtmlNode();
28.                 pNode->type = NODE_CONTENT;
```

为解决易语言程序被杀毒软件误报而进行的一... (19)

网站友情链接

a.farproc.comEF官方网站

BLOG友情链接

骆新发言的BLOGfarproc's blogmonkeycz's blogEF官方博客 (RSS)陈皓专栏 (RSS)bywayboy's blog (RSS)石年遗梦(jouby) (RSS)

```
29.         pNode->text = duplicateStrUtil1(s, L'<', true);
30.     }
31.     s = p + 1;
32. }
33. else if(c == L'>')
34. {
35.     if(p > s)
36.     {
37.         //Add HtmlTag Node
38.         pNode = NewHtmlNode();
39.         while(isspace(*s)) s++;
40.         pNode->type = (*s != L'/' ? NODE_START_TAG : NODE_CLOSE_TAG);
41.         if(*s == L'/' ) s++;
42.         copyStrUtil1(pNode->tagName, MAX_HTML_TAG_LENGTH, s, L'>', true);
43.         //处理自闭闭的结点，如 <br/>, 删除tagName中可能会有的'/'字符
44.         //自闭闭的结点的type设置为NODE_START_TAG应该可以接受(否则要引入新的
NODE_STARTCLOSE_TAG)
45.         int tagNamelen = wcslen(pNode->tagName);
46.         if(pNode->tagName[tagNamelen-1] == L'/')
47.             pNode->tagName[tagNamelen-1] = L'/0';
48.         //处理结点属性
49.         for(int i = 0; i < tagNamelen; i++)
50.         {
51.             if(pNode->tagName[i] == L' ' //第一个空格后面跟的是属性列表
52.             || pNode->tagName[i] == L'=' ) //扩展支持这种格
式: <tagName=value>, 等效于<tagName tagName=value>
53.             {
54.                 WCHAR* props = (pNode->tagName[i] == L' ' ? s + i + 1 : s);
55.                 pNode->text = duplicateStrUtil1(props, L'>', true);
56.                 int nodeTextLen = wcslen(pNode->text);
57.                 if(pNode->text[nodeTextLen-1] == L'/') //去掉最后可能会有的'/'字
符, 如这种情况: 
58.                     pNode->text[nodeTextLen-1] = L'/0';
59.                 pNode->tagName[i] = L'/0';
60.                 parseNodeProps(pNode); //parse props
61.                 break;
62.             }
63.         }
64.         pNode->tagType = getHtmlTagTypeFromName(pNode->tagName);
65.     }
66.     s = p + 1;
67. }
68. p++;
69. }
70. if(p > s)
71. {
72.     //Add Text Node
73.     pNode = NewHtmlNode();
74.     pNode->type = NODE_CONTENT;
75.     pNode->text = duplicateStr(s, -1);
76. }
77. #ifdef _DEBUG
78.     dumpHtmlNodes(); //just for test
79. #endif
80. }
```

下面是负责解析“开始标签”属性表文本（形如“key1=value1 key2=value2 key3”）的代码，parseNodeProps()，核心思路是按空格和等号字符进行分割属性名和属性值，由于想兼容HTML4.01及以前的不标准的属性表写法（如没有=号也没有属性值），颇费周折：

[cpp]

```
01. //virtual
02. void HtmlParser::parseNodeProps(HtmlNode* pNode)
03. {
04.     if(pNode == NULL || pNode->propCount > 0 || pNode->text == NULL)
05.         return;
06.     WCHAR* p = pNode->text;
07.     WCHAR *ps = NULL;
08.     CMem mem;
09.     bool inQuote1 = false, inQuote2 = false;
10.     WCHAR c;
11.     while(c = *p)
12.     {
13.         if(c == L'/' )
```

```

14.         {
15.             inQuote1 = !inQuote1;
16.         }
17.         else if(c == L'/'')
18.         {
19.             inQuote2 = !inQuote2;
20.         }
21.         if(!inQuote1 && !inQuote2) && (c == L' ' || c == L'/t' || c == L'='))
22.         {
23.             if(ps)
24.             {
25.                 mem.AddPointer(duplicateStrAndUnquote(ps, p - ps));
26.                 ps = NULL;
27.             }
28.             if(c == L'='')
29.                 mem.AddPointer(NULL);
30.         }
31.         else
32.         {
33.             if(ps == NULL)
34.                 ps = p;
35.         }
36.         p++;
37.     }
38.     if(ps)
39.         mem.AddPointer(duplicateStrAndUnquote(ps, p - ps));
40.     mem.AddPointer(NULL);
41.     mem.AddPointer(NULL);
42.     WCHAR** pp = (WCHAR**) mem.GetPtr();
43.     CMem props;
44.     for(int i = 0, n = mem.GetSize() / sizeof(WCHAR*) - 2; i < n; i++)
45.     {
46.         props.AddPointer(pp[i]); //prop name
47.         if(pp[i+1] == NULL)
48.         {
49.             props.AddPointer(pp[i+2]); //prop value
50.             i += 2;
51.         }
52.         else
53.             props.AddPointer(NULL); //prop vvalue
54.     }
55.     pNode->propCount = props.GetSize() / sizeof(WCHAR*) / 2;
56.     pNode->props = (HtmlNodeProp*) props.Detach();
57. }

```

根据标签名称取标签类型的getHtmlTagTypeFromName()方法，就非常直白了，查表，逐一识别：



```

[cpp]
01. //virtual
02. HtmlTagType HtmlParser::getHtmlTagTypeFromName(const WCHAR* szTagName)
03. {
04.     //todo: uses hashmap
05.     struct N2T { const WCHAR* name; HtmlTagType type; };
06.     static N2T n2tTable[] =
07.     {
08.         { L"A", TAG_A },
09.         { L"FONT", TAG_FONT },
10.         { L"IMG", TAG_IMG },
11.         { L"P", TAG_P },
12.         { L"DIV", TAG_DIV },
13.         { L"SPAN", TAG_SPAN },
14.         { L"BR", TAG_BR },
15.         { L"B", TAG_B },
16.         { L"I", TAG_I },
17.         { L"HR", TAG_HR },
18.     };
19.     for(int i = 0, count = sizeof(n2tTable)/sizeof(n2tTable[0]); i < count; i++)
20.     {
21.         N2T* p = &n2tTable[i];
22.         if(wcsicmp(p->name, szTagName) == 0)
23.             return p->type;
24.     }
25.     return TAG_UNKNOWN;
26. }

```

请注意，上文负责解析属性表的`parseNodeProps()`函数，和负责识别标签名称的`getHtmlTagTypeFromName()`函数，都是虚函数（`virtual method`）。我(liigo)这么设计是有深意的，给使用者留下了很大的定制空间，可以自由发挥。例如，通过在子类中覆盖/覆写（`override`）`parseNodeProps()`方法，可以采用更好的解析算法，或者干脆不做任何处理以提高HTML解析效率——将来某一时间可以调用基类同名函数专门解析特定标签的属性表；例如，通过在子类中覆盖/覆写（`override`）`getHtmlTagTypeFromName()`方法，使用者可以选择识别跟多的标签名称（包括自定义标签），或者识别更少的标签名称，甚至不识别任何标签名称（以便提高解析效率）。以编写网络爬虫程序为实例，它多数情况下通常只需识别`<A>`标签及其属性就足够了，没必要浪费CPU运算去识别其它标签、解析其他标签属性。

至于HTML文本解析器的用途，我目前想到的有：用于HTML格式检查或规范化，用于重新排版HTML文本，用于编写网络爬虫程序/搜索引擎，用于基于HTML模板的动态网页生成，用于HTML网页渲染前的基础解析，等等。

下面附上完整源码，仅供参考，欢迎指正。

HtmlParser.h :

[cpp:collapse]

HtmlParser.cpp :

[cpp:collapse]

全文完，谢谢。

2011-1-22 liigo 补记：本文所提供的源代码，目前有未完善之处，如没有考虑到内嵌JavaScript代码和HTML注释中的特殊字符（特别是尖括号）对解析器的影响，另外还可能有其他疏漏和bug，故代码仅可用于学习参考研究使用。我今后也将继续改进此HTML语法解析器。特此声明。

2012-5-5 liigo 补记：在刚刚过去的半个多月里，我又对此HTML解析器做了很多改进（并将持续改进），目前应该说是比较成熟和完善了。源代码已经放到GitHub：<https://github.com/liigo/html-parser>。另外，本文嵌入的代码已经很旧了（且其中C/C++转义字符被CSDN博客系统粗暴替换），但主要的设计和实现思路依然有效。我也有计划新写一篇本文的2.0版。

上一篇：liigo：爱可视70平板电脑使用感受，遗憾与满足并存

下一篇：使用天乐软件加密狗(JDProtect)保护您的软件，防止程序被跟踪/逆向/反编译/破解

分享到：



查看评论

43楼 litongli1 2012-04-24 11:12发表



你好，能不能发个完整的源程序

Re: [liigo](#) 2012-05-04 09:39发表



回复 [litongli1](#)：参见文章末尾的补记

42楼 [litongli1](#) 2012-04-24 10:58发表



你好 请问后面源码中HtmlParser.h文件中的#include "common.h" 中的 common.h是什么 在哪 能否给出来 非常感谢

41楼 [shuirh](#) 2011-12-12 17:06发表



我感觉用编译原理写出来的会很麻烦, 我喜欢轻量级的.

Re: [liigo](#) 2011-12-16 20:03发表



回复 [shuirh](#)：同意

40楼 [Maxwellcome](#) 2011-11-07 20:14发表



作者的思想不是SAX吗？在轻量级的前提下，SAX处理HTML页面在某些方面确实优于DOM，不过DOM可以保留所有Node的结构化信息，比如其父亲、孩子、兄弟节点等，而SAX就力有不逮了。至于作者说的“稍加处理即可形成属性层次节点”，不知道这个稍加处理能达到怎样的性能啊。

Re: [liigo](#) 2011-12-16 20:09发表



回复 [Maxwellcome](#)：必要的时候，在流线型节点的基础上自行生成DOM树形层次，是比较容易而且直观的。例如，如果前面一个节点还没有关闭，以后的节点就是当前未关闭节点的子节点。

39楼 [zuoruixiaoren](#) 2011-10-26 10:08发表



不得不佩服楼主.....

38楼 [walker1985](#) 2011-10-20 10:26发表



我是不太擅长这方面的东西。学习 学习！

37楼 [bywayboy](#) 2011-10-19 00:08发表



我倒是学过编译原理，也擅长写这种东西。大学就写过语法高亮的编辑器。再到后来乱七八糟的json之类的。代码如其人，看liigo的代码就是一种享受。
编程有三种人，一种写给人看的，那是教师。一种写个计算机看的，那是骨灰级程序员。一种两者都能看的，那是大师。

36楼 [cygwinner](#) 2011-04-16 21:32发表



用正则表达式不是更精简？

35楼 [SGPRO](#) 2011-02-19 14:04发表



博主还没学过编译技术吧，慢慢来，大三就开课了。

Re: [chs15](#) 2011-10-17 09:19发表



回复 [SGPRO](#)：动不动就编译原理, LZ又不是要实现一个脚本库...

本来就是轻量级的需求;

34楼 [beyondljs](#) 2011-02-17 13:36发表



这种文章也“推荐”？？有没有学过编译原理？？

Re: [liigo](#) 2011-04-18 15:00发表



回复 [beyondljs](#)：你用你学过的编译原理，来写个相同功能的代码，看看能不能比这个更简单更好用。抱着书不放的都是书呆子；放下书，写写代码。

33楼 [ychenzj](#) 2011-02-14 14:53发表



还可以，值得一看

32楼 [Cody_Yu](#) 2011-02-12 11:33发表



如果html不太正规，比如没有使用结束标签怎么办？而如果严格了，那不就成了xml了么？

31楼 [NetSniffer](#) 2011-02-11 23:18发表



开源的Htmlparser以前用过，不过效率不是很高

Re: [liigo](#) 2011-04-18 15:05发表



回复 **NetSniffer**：我这个号称重复发明轮子，如果以前有人写过这样的代码也很正常，我一普通人能想到的，肯定以前有人也能想到。我重复发明轮子意思，就是用自己的思路自己的代码，锻炼自己。

30楼 **Dancer_20080215** 2011-02-11 17:00发表



好像是抄袭哪个程序的？那程序员是德国人，该代码1998年就发布过的。

29楼 **isp_jlu** 2011-02-11 16:24发表



不错不错

28楼 **lwyx2000** 2011-02-11 10:01发表



喜欢这样的文章，只会调用没什么意思

27楼 **jyl_sh** 2011-02-11 09:06发表



你的Common.h文件呢？

26楼 **zhouhonggnay** 2011-02-09 20:14发表



好强大哈。。。。。

25楼 **yzaaa** 2011-01-30 23:04发表



凡是做基础研究的国人,都应该支持,我们太缺这样的人才了,重复发明轮子不是坏事,因为我们连轮子都造不出来

24楼 **mxlinux** 2011-01-27 20:13发表



凡是做基础研究的国人,都应该支持,我们太缺这样的人才了,重复发明轮子不是坏事,因为我们连轮子都造不出来

23楼 **shendl** 2011-01-27 14:22发表



Java+正则表达式的实现：HtmlCleaner。介绍：<http://blog.csdn.net/shendl/archive/2009/10/27/4735160.aspx>
楼主看看这个东东吧。

22楼 **livemylife** 2011-01-27 11:54发表



这个还是LEX YACC写一下比较给力。
不过HTML的不规范和容错是大头

21楼 **ivension** 2011-01-25 13:38发表



很好的东西，收藏了。



20楼 **alivio** 2011-01-25 12:53发表



大哥，解析统一规范的html文档谁不会呀，关键是纠错，容错，以及兼容不同的规范。

Re: **liigo** 2011-01-26 00:40发表



回复 **alivio**：我也想写的完善一下呀，大家多提具体的改进意见嘛。目前已经尽量兼容不规范的html了，如：不关闭的节点、未用引号括住的属性值、没有属性值的属性等。此外还有对规范的扩充语法：<color=red>。

19楼 **shijf2010** 2011-01-25 11:00发表



博主，你这个几乎看不到C++的东西，是C语言的吧？要用C++的话，用boost.spirit几句话就搞定了，boost.spirit自带的例子就有XML解析和微型C语言解析器。

Re: **liigo** 2011-01-26 00:29发表



回复 **shijf2010**：恩，C++特性我基本只用类、构造析构函数、随时定义变量、//注释等。调用别人的代码总归不是自己的，自己动手才能丰衣足食，我这个“重复发明轮子”系列自有道理，有机会写出来跟大家分享。

18楼 **networkwx** 2011-01-25 09:51发表



我以为是有渲染部分！不过还是谢谢楼主。。

Re: **liigo** 2011-01-26 00:32发表



回复 **networkwx**：我以这个html解析器代码为基础，做了一个简单的html渲染引擎，但是不方便公开全部代码，过一段时间也许可能会说一说开发思路。谢谢支持。

17楼 **alloycn** 2011-01-24 18:43发表



请支持,CSDN经常扣分下载不了,没有分了.谢谢



16楼 **wenhui4564455** 2011-01-24 18:07发表



15楼 shendl 2011-01-24 15:28发表



用正则表达式解析HTML是最好的方法。已经有事先。
Html解析以后应该形成一颗单根树。

Re: liigo 2011-01-26 00:21发表



回复 shendl：哇，从此另眼相看正则表达式了。

14楼 lishuai369 2011-01-23 17:08发表



用Boost：：spirit库很好实现，我都可以写个C++代码解析器，html和xml差不多，网上到处都是

Re: liigo 2011-01-26 00:42发表



回复 lishuai369：调用别人的代码总归不是自己的，自己动手才能丰衣足食，我这个“重复发明轮子”系列自有道理，有机会写出来跟大家分享。

13楼 b2b160 2011-01-22 18:13发表



思路不错,不过Node干吗不用继承呢?用继承就不会有冗余吧?
此外duplicateStrUtil这个也许会导致效率比较低的,可以参考SAX的做法.那样可能高效点.
还有就是HTML的层级关系还是比较基本的属性,应该不好废弃的.

Re: liigo 2011-01-22 20:58发表



回复 b2b160：用了继承后，不同类型的Node的sizeof就不一样了，我就不能用一个内存数组存储所有Node了，要么存对象（频繁复制对象）要么存指针（频繁创建对象）。总之差别不是很明显，都可选用的。

Re: liigo 2011-01-22 20:50发表



回复 b2b160：说的好。duplicateStrUtil这个会导致效率比较低，但在不知道何时结束拷贝的情况下，似乎只能逐个字符复制；SAX的做法我目前不太清楚，还望指教。至于HTML的层级关系，我没有废弃它，只是没有实现它，只要外面再封装一层就可以轻松拥有层级关系了。我的意思是说，html核心语法规则，与html节点层次关系，是两个层面的任务，可以分开独立实现，后者以前者为基础。

12楼 yangboshi_2010 2011-01-22 11:24发表



一学期的时间，HTML荒废了，这都看不懂

11楼 hxx_7 2011-01-22 10:45发表



读了博主的文章对我启发很大,很想学习关于HTML文本解析的思路方法,向你学习,谢谢
我没接触过HTML文本解析方面的,所以对代码中有些成员和调用不是很清楚,问下楼主CMem m_HtmlNodes;CString m_html;成员是自己封装的类型吗,还有里面的函数调用例如:MFreeMemory等,我想编译调试.
我是用VC++建立的win32控制台项目,不知道是否正确.
希望博主帮我解答下,再次感谢

Re: liigo 2011-01-22 13:31发表



回复 hxx_7：CMem是一个自定义的简单的内存缓冲设施,用std::vector替换也完全是可行的.因不是本文重点,才没有涉及.

10楼 aofengdaxia 2011-01-22 09:00发表



9楼 leesao0802 2011-01-22 00:09发表



8楼 solo_coder 2011-01-21 11:32发表



.....
这个，为毛不用语法/词法生成器来做？

Re: liigo 2011-01-21 18:58发表



回复 solo_coder：欢迎提供更好的实现方案，有具体代码更好，好像学习交流嘛

7楼 leonliu7558168 2011-01-21 10:37发表



希望楼主多发此类学习文章！此乃行大善啊！

Re: liigo 2011-01-21 18:50发表



回复 leonliu7558168：代码共享，互相学习嘛，谢谢支持

- 6楼 [songzhu](#) 2011-01-21 09:56发表
-  学习用还是很好的。
- Re: [liigo](#) 2011-01-22 00:57发表
-  回复 [songzhu](#)：还很实用呢，我写这个可不是为了好玩，是准备拿来干正事的，以后应该还会通过博客提供应用实例，以便进一步检验（或证明）此html语法解析器的实用价值。
- 5楼 [zzymusic](#) 2011-01-21 09:19发表
-  很好 学习了
- 4楼 [jszj](#) 2011-01-21 08:59发表
-  不赖，学习
- 3楼 [DYFDWX](#) 2011-01-21 08:48发表
-  
- 2楼 [dragon32](#) 2011-01-20 12:28发表
-  不錯。但這不就掉去了層級關係的資料嗎?? 我想可加個 **stack** 比較好？只是我的一點淺見。
- Re: [liigo](#) 2011-01-21 18:48发表
-  回复 [dragon32](#)：是的，此html语法解析器输出的是线性节点数组，但也是可扩展的，用户代码引入一个节点栈稍加处理即可形成树形层次节点，比较容易。
- 1楼 [hans_netizen](#) 2011-01-20 12:04发表
-  能否对比一下其他解析器(如Webkit), 主要差别和优势，谢谢！
- Re: [liigo](#) 2011-01-21 18:40发表
-  回复 [hans_netizen](#)：所谓杀鸡焉用牛刀，**webkit**有点太庞大了，本文提供的是我自行设计编码的轻量级html语法解析器，自我感觉功能上也不是太弱，日常够用。

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