

MediaInfo源代码分析 4：Inform()函数

2013年10月08日 21:40:09 阅读数：4006

MediaInfo源代码分析系列文章列表：

[MediaInfo源代码分析 1：整体结构](#)

[MediaInfo源代码分析 2：API函数](#)

[MediaInfo源代码分析 3：Open\(\)函数](#)

[MediaInfo源代码分析 4：Inform\(\)函数](#)

[MediaInfo源代码分析 5：JPEG解析代码分析](#)

我们来看一下MediaInfo中的Inform()函数的内部调用过程

首先Inform()函数封装了MediaInfo_Internal类中的Inform()函数

```
[cpp]
1. //返回文件信息
2. String MediaInfo::Inform(size_t)
3. {
4.     //封装了一层
5.     return Internal->Inform();
6. }
```

查看一下MediaInfo_Internal类中的Inform()函数的源代码：

```
[cpp]
1. // 获取信息
2. Zstring MediaInfo_Internal::Inform()
3. {
4.     CS.Enter();
5.     if (Info && Info->Status[File_Analyze::IsUpdated])
6.         Info->Open_Buffer_Update();
7.     CS.Leave();
8.
9.     if (MediaInfoLib::Config.Inform_Get()==_T("MPEG-7"))
10.        return Export_Mpeg7().Transform(*this);
11.     if (MediaInfoLib::Config.Inform_Get()==_T("PBCore") || MediaInfoLib::Config.Inform_Get()==_T("PBCore_1.2"))
12.        return Export_PBCore().Transform(*this);
13.     if (MediaInfoLib::Config.Inform_Get()==_T("reVTMD"))
14.        return _T("reVTMD is disabled due to its non-free licensing."); //return Export_reVTMD().Transform(*this);
15.     //获取相应的信息
16.     if (!(
17.         MediaInfoLib::Config.Inform_Get(__T("General")).empty()
18.         && MediaInfoLib::Config.Inform_Get(__T("Video")).empty()
19.         && MediaInfoLib::Config.Inform_Get(__T("Audio")).empty()
20.         && MediaInfoLib::Config.Inform_Get(__T("Text")).empty()
21.         && MediaInfoLib::Config.Inform_Get(__T("Chapters")).empty()
22.         && MediaInfoLib::Config.Inform_Get(__T("Image")).empty()
23.         && MediaInfoLib::Config.Inform_Get(__T("Menu")).empty()
24.     ))
25.     {
26.         //获取各种信息
27.         //Retour即为返回的字符串
28.         Zstring Retour;
29.         Retour+=MediaInfoLib::Config.Inform_Get(__T("File_Begin"));
30.         Retour+=MediaInfoLib::Config.Inform_Get(__T("General_Begin"));
31.         Retour+=Inform(Stream_General, 0, false);
32.         Retour+=MediaInfoLib::Config.Inform_Get(__T("General_End"));
33.         if (Count_Get(Stream_Video))
34.             Retour+=MediaInfoLib::Config.Inform_Get(__T("Video_Begin"));
35.         for (size_t I1=0; I1<Count_Get(Stream_Video); I1++)
36.         {
37.             Retour+=Inform(Stream_Video, I1, false);
38.             if (I1!=Count_Get(Stream_Video)-1)
39.                 Retour+=MediaInfoLib::Config.Inform_Get(__T("Video_Middle"));
40.         }
41.         if (Count_Get(Stream_Video))
42.             Retour+=MediaInfoLib::Config.Inform_Get(__T("Video_End"));
43.         if (Count_Get(Stream_Audio))
44.             Retour+=MediaInfoLib::Config.Inform_Get(__T("Audio_Begin"));
45.         for (size_t I1=0; I1<Count_Get(Stream_Audio); I1++)
46.         {
47.             Retour+=Inform(Stream_Audio, I1, false);
48.             if (I1!=Count_Get(Stream_Audio)-1)
```

```

48.         if (I1!=Count_Get(Stream_Audio)-1)
49.             Retour+=MediaInfoLib::Config.Inform_Get(__T("Audio_Middle"));
50.     }
51.     if (Count_Get(Stream_Audio))
52.         Retour+=MediaInfoLib::Config.Inform_Get(__T("Audio_End"));
53.     if (Count_Get(Stream_Text))
54.         Retour+=MediaInfoLib::Config.Inform_Get(__T("Text_Begin"));
55.     for (size_t I1=0; I1<Count_Get(Stream_Text); I1++)
56.     {
57.         Retour+=Inform(Stream_Text, I1, false);
58.         if (I1!=Count_Get(Stream_Text)-1)
59.             Retour+=MediaInfoLib::Config.Inform_Get(__T("Text_Middle"));
60.     }
61.     if (Count_Get(Stream_Text))
62.         Retour+=MediaInfoLib::Config.Inform_Get(__T("Text_End"));
63.     if (Count_Get(Stream_Other))
64.         Retour+=MediaInfoLib::Config.Inform_Get(__T("Chapters_Begin"));
65.     for (size_t I1=0; I1<Count_Get(Stream_Other); I1++)
66.     {
67.         Retour+=Inform(Stream_Other, I1, false);
68.         if (I1!=Count_Get(Stream_Other)-1)
69.             Retour+=MediaInfoLib::Config.Inform_Get(__T("Chapters_Middle"));
70.     }
71.     if (Count_Get(Stream_Other))
72.         Retour+=MediaInfoLib::Config.Inform_Get(__T("Chapters_End"));
73.     if (Count_Get(Stream_Image))
74.         Retour+=MediaInfoLib::Config.Inform_Get(__T("Image_Begin"));
75.     for (size_t I1=0; I1<Count_Get(Stream_Image); I1++)
76.     {
77.         Retour+=Inform(Stream_Image, I1, false);
78.         if (I1!=Count_Get(Stream_Image)-1)
79.             Retour+=MediaInfoLib::Config.Inform_Get(__T("Image_Middle"));
80.     }
81.     if (Count_Get(Stream_Image))
82.         Retour+=MediaInfoLib::Config.Inform_Get(__T("Image_End"));
83.     if (Count_Get(Stream_Menu))
84.         Retour+=MediaInfoLib::Config.Inform_Get(__T("Menu_Begin"));
85.     for (size_t I1=0; I1<Count_Get(Stream_Menu); I1++)
86.     {
87.         Retour+=Inform(Stream_Menu, I1, false);
88.         if (I1!=Count_Get(Stream_Menu)-1)
89.             Retour+=MediaInfoLib::Config.Inform_Get(__T("Menu_Middle"));
90.     }
91.     if (Count_Get(Stream_Menu))
92.         Retour+=MediaInfoLib::Config.Inform_Get(__T("Menu_End"));
93.     Retour+=MediaInfoLib::Config.Inform_Get(__T("File_End"));
94.     //可以在此加入视频质量检测-----
95.
96.     //字符串替换？各种换行符统统改为"\n"-----
97.     Retour.FindAndReplace(__T("\\r\\n"), __T("\n"), 0, Ztring_Recursive);
98.     Retour.FindAndReplace(__T("\\r"), __T("\n"), 0, Ztring_Recursive);
99.     Retour.FindAndReplace(__T("\\n"), __T("\n"), 0, Ztring_Recursive);
100.    Retour.FindAndReplace(__T("\r\n"), __T("\n"), 0, Ztring_Recursive);
101.    Retour.FindAndReplace(__T("\r"), __T("\n"), 0, Ztring_Recursive);
102.    Retour.FindAndReplace(__T("\n"), MediaInfoLib::Config.LineSeparator_Get(), 0, Ztring_Recursive);
103.
104.    //Special characters
105.    Retour.FindAndReplace(__T("|SC1|"), __T("\\"), 0, Ztring_Recursive);
106.    Retour.FindAndReplace(__T("|SC2|"), __T("["), 0, Ztring_Recursive);
107.    Retour.FindAndReplace(__T("|SC3|"), __T("]"), 0, Ztring_Recursive);
108.    Retour.FindAndReplace(__T("|SC4|"), __T(", "), 0, Ztring_Recursive);
109.    Retour.FindAndReplace(__T("|SC5|"), __T(";"), 0, Ztring_Recursive);
110.    Retour.FindAndReplace(__T("|SC6|"), __T("("), 0, Ztring_Recursive);
111.    Retour.FindAndReplace(__T("|SC7|"), __T(")"), 0, Ztring_Recursive);
112.    Retour.FindAndReplace(__T("|SC8|"), __T("("), 0, Ztring_Recursive);
113.    Retour.FindAndReplace(__T("|SC9|"), __T(", "), 0, Ztring_Recursive);
114.
115.    return Retour;
116. }
117.
118. //Informations
119. Ztring Retour;
120. bool HTML=false;
121. bool XML=false;
122. bool CSV=false;
123. //获取配置信息（输出格式）
124. if (MediaInfoLib::Config.Inform_Get()==__T("HTML"))
125.     HTML=true;
126. if (MediaInfoLib::Config.Inform_Get()==__T("XML"))
127.     XML=true;
128. if (MediaInfoLib::Config.Inform_Get()==__T("CSV"))
129.     CSV=true;
130.
131. if (HTML) Retour+=__T("<html>\n\n<head>\n\n<META http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\" /></head>\n\n<body>\n\n");
132. if (XML) Retour+=__T("<File>\n\n");
133.
134. for (size_t StreamKind=(size_t)Stream_General; StreamKind<Stream_Max; StreamKind++)
135. {
136.     //Pour chaque type de flux
137.     for (size_t StreamPos=0; StreamPos<(size_t)Count_Get((stream_t)StreamKind); StreamPos++)
138.     {

```

```

139.         //Pour chaque stream
140.         //输出为HTML
141.         if (HTML) Retour+= __T("
<table width=\"100%\" border=\"0\" cellpadding=\"1\" cellspacing=\"2\" style=\"border:1px solid Navy\">\n<tr>\n    <td width=\"150\">
<h2>");
142.         //输出为XML
143.         if (XML) Retour+= __T("<track type=\"");
144.         Ztring A=Get((stream_t)StreamKind, StreamPos, __T("StreamKind/String"));
145.         Ztring B=Get((stream_t)StreamKind, StreamPos, __T("StreamKindPos"));
146.         if (!XML && !B.empty())
147.         {
148.             if (CSV)
149.                 A+=__T(",");
150.             else
151.                 A+=MediaInfoLib::Config.Language_Get(__T(" Config_Text_NumberTag"));
152.             A+=B;
153.         }
154.         Retour+=A;
155.         if (XML)
156.         {
157.             Retour+= __T("\"");
158.             if (!B.empty())
159.             {
160.                 Retour+= __T(" streamid=\"");
161.                 Retour+=B;
162.                 Retour+= __T("\"");
163.             }
164.         }
165.         //输出为HTML
166.         if (HTML) Retour+= __T("</h2></td>\n </tr>");
167.         //输出为XML
168.         if (XML) Retour+= __T(">");
169.         Retour+=MediaInfoLib::Config.LineSeparator_Get();
170.         Retour+=Inform((stream_t)StreamKind, StreamPos, false);
171.         Retour.FindAndReplace(__T("\\"), __T("|SC1|"), 0, Ztring_Recursive);
172.         if (HTML) Retour+= __T("</table>\n<br />");
173.         if (XML) Retour+= __T("</track>\n");
174.         Retour+=MediaInfoLib::Config.LineSeparator_Get();
175.     }
176. }
177. //输出为HTML
178. if (HTML) Retour+= __T("\n</body>\n</html>\n");
179. //输出为XML
180. if (XML) Retour+= __T("</File>\n");
181. //字符串替换？
182. Retour.FindAndReplace(__T("\\r\\n"), __T("\n"), 0, Ztring_Recursive);
183. Retour.FindAndReplace(__T("\\r"), __T("\n"), 0, Ztring_Recursive);
184. Retour.FindAndReplace(__T("\\n"), __T("\n"), 0, Ztring_Recursive);
185. Retour.FindAndReplace(__T("\\r\\n"), __T("\n"), 0, Ztring_Recursive);
186. Retour.FindAndReplace(__T("\\r"), __T("\n"), 0, Ztring_Recursive);
187. Retour.FindAndReplace(__T("\n"), MediaInfoLib::Config.LineSeparator_Get(), 0, Ztring_Recursive);
188.
189. //Special characters
190. Retour.FindAndReplace(__T("|SC1|"), __T("\\"), 0, Ztring_Recursive);
191. Retour.FindAndReplace(__T("|SC2|"), __T("("), 0, Ztring_Recursive);
192. Retour.FindAndReplace(__T("|SC3|"), __T("]"), 0, Ztring_Recursive);
193. Retour.FindAndReplace(__T("|SC4|"), __T(","), 0, Ztring_Recursive);
194. Retour.FindAndReplace(__T("|SC5|"), __T(";"), 0, Ztring_Recursive);
195. Retour.FindAndReplace(__T("|SC6|"), __T("("), 0, Ztring_Recursive);
196. Retour.FindAndReplace(__T("|SC7|"), __T(")"), 0, Ztring_Recursive);
197. Retour.FindAndReplace(__T("|SC8|"), __T(")"), 0, Ztring_Recursive);
198. Retour.FindAndReplace(__T("|SC9|"), __T(","), 0, Ztring_Recursive);
199. return Retour;
200. }

```

函数比较复杂，从代码中我们可以看出，Inform()的实质还是使用Get()一个一个取出所有的属性值。

当指定输出为XML或者是HTML的时候，在输出的字符串上加上相应的标签（例如，输出为HTML的时候，字符串每一行上加上“</tr><tr>”，首尾加上“<table></table>”）

具体每一块代码的含义已经写在注释中了。

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/12451561>

文章标签： [源代码](#) [MediaInfo](#) [Inform](#)

个人分类： [MediaInfo](#)

所属专栏： [开源多媒体项目源代码分析](#)

