在 TinyXML 中，根据 XML 的各种元素来定义了一些类：

TiXmlBase：整个 TinyXML 模型的基类。

TiXmlAttribute：对应于 XML 中的元素的属性。

TiXmlNode：对应于 DOM 结构中的节点。

TiXmlComment：对应于 XML 中的注释。

TiXmlDeclaration：对应于 XML 中的申明部分，即<？versiong="1.0"?>。

TiXmlDocument：对应于 XML 的整个文档。

TiXmlElement：对应于 XML 的元素。

TiXmlText：对应于 XML 的文字部分。

TiXmlUnknown：对应于 XML 的未知部分。

TiXmlHandler：定义了针对 XML 的一些操作。

例如:

```
<?xml version="1.0" standalone=no>
<!– Our to do list data –>
<ToDo>
<Item priority="1"> Go to the <bold>Toy store!</bold></Item>
<Item priority="2"> Do bills</Item>
</ToDo>
```

整个对象树：

TiXmlDocument "demo.xml"

TiXmlDeclaration "version=’1.0′" "standalone=no"

TiXmlComment " Our to do list data"

TiXmlElement "ToDo"

TiXmlElement "Item" Attribtutes: priority = 1

TiXmlText "Go to the "

TiXmlElement "bold"

TiXmlText "Toy store!"

TiXmlElement "Item" Attributes: priority=2

TiXmlText "Do bills"

在 tinyXML 中,用 FirstChild（"名字"）查找节点时,调用 FirstChild 函数的节点与要查找的节点必须成“父子关系”。

句柄

想要健壮地读取一个 XML 文档，检查方法调用后的返回值是否为 null 是很重要的。一种安全的检错实现可能会产生像这样的代码：

```
TiXmlElement* root = document.FirstChildElement( "Document" );
if ( root )
{
TiXmlElement* element = root->FirstChildElement( "Element" );
if ( element )
{
TiXmlElement* child = element->FirstChildElement( "Child" );
if ( child )
{
TiXmlElement* child2 = child->NextSiblingElement( "Child" );
if ( child2 )
{
// Finally do something useful.
```

用句柄的话就不会这么冗长了，使用 TiXmlHandle 类，前面的代码就会变成这样：

```
 TiXmlHandle  docHandle(  &document  );
 TiXmlElement*  child2  =  docHandle.FirstChild(  "Document"  ).FirstChild(  "Elem
ent"  ).Child(  "Child",  1  ).ToElement();
 if  (  child2  )
{
// do something useful
```

一、读取 XML,设置节点文本

如下 XML 片段:

```
 <?xml  version="1.0"  encoding="UTF-8"  standalone="yes"  ?>
 <ZXML>
        <ZAPP>
                <VBS_RUNTIME_PARAMS>
                        <BROADCAST_VERSION  info="版本">8</BROADCAST_VERSIO
N>
                        <Broadcast>
                          <FileCount  info="资源文件个数">69</FileCount>
                          <SOURCE_1>
                                <ID  info="图片编号">1</ID>
                                <Version  info="图片版本">1</Version>
                                <Path  info="图片路径">/mnt/share/1.bmp</
Path>
                                <FileMode  info="文件处理模式">0</FileMod
e>
```

```
                                    </SOURCE_1>
                                    <SOURCE_2>
                                            <Path  info="图片路径">/mnt/share/2.bmp</
Path>
                                            <ID  info="图片编号">2</ID>
                                            <Version  info="图片版本">1</Version>
                                            <FileMode  info="文件处理模式">0</FileMod
e>
                                    </SOURCE_2>
                                    ……
                                    </Broadcast>
                        </VBS_RUNTIME_PARAMS>
                </ZAPP>
 </ZXML>
```

要设置BROADCAST_VERSION节点的值 8为其他值,可参考如下代码(将值加1):

用ReplaceChild(TiXmlNode* replaceThis, const TiXmlNode& withThis)方法替换

```
        TiXmlDocument  doc("zapp.conf");
         doc.LoadFile();
         TiXmlHandle  docHandle(  &doc  );
 TiXmlElement*  Broadcast_ver  =  docHandle.FirstChild("ZXML").FirstChild("ZAPP").
FirstChild("VBS_RUNTIME_PARAMS").FirstChildElement("BROADCAST_VERSION").ToElement();
         TiXmlNode  *  oldnode  =    Broadcast_ver->FirstChild();
        const  char  *ver  =  Broadcast_ver->GetText();
         int  oldVer  =  atoi(ver);
         CString  newVer;
         newVer.Format("%d",oldVer+1);
         TiXmlText  newText(newVer);
```

```
        Broadcast_ver->ReplaceChild(oldnode,newText);
        AfxMessageBox(Broadcast_ver->GetText());//输出值
        doc.SaveFile();
```

二,删除节点,属性值

RemoveChild(TiXmlNode* removeThis)方法删除父节点的子节点,

RemoveAttribute(const char * name)方法删除属性值.

例如删除 BROADCAST_VERSION 节点

```
 TiXmlHandle  docHandle(  &doc  );
        TiXmlElement*  Broadcast_ver  =  docHandle.FirstChild("ZXML").FirstChild("ZAPP").FirstChild("VBS_RUNTIME_PARAMS").ToElement();

        TiXmlNode  *  node  =    Broadcast_ver->FirstChild("BROADCAST_VERSION");

        Broadcast_ver->RemoveChild(node);
```

也可以删除整个 SOURCE_1 节点:

```
 TiXmlHandle  docHandle(  &doc  );
        TiXmlElement*  Broadcast  =  docHandle.FirstChild("ZXML").FirstChild("ZAPP").FirstChild("VBS_RUNTIME_PARAMS").FirstChild("Broadcast").ToElement();

        TiXmlNode  *  node  =    Broadcast->FirstChild("SOURCE_1");

        Broadcast->RemoveChild(node);
```

删除 BROADCAST_VERSION 的 info 属性:

```
 TiXmlHandle  docHandle(  &doc  );
        TiXmlElement*  Broadcast_ver  =  docHandle.FirstChild("ZXML").FirstChild
("ZAPP").FirstChild("VBS_RUNTIME_PARAMS").FirstChildElement("BROADCAST_VERSION").To
Element();

        Broadcast_ver->RemoveAttribute("info");  //删除 info
```

可以借助 NextSiblingElement()方法实现递归删除.

三,添加节点,属性值

例如在 SOURCE_3 下添加 BROADCAST_PID 节点:

```
 TiXmlHandle  docHandle(  &doc  );
        TiXmlElement*  Broadcast  =  docHandle.FirstChild("ZXML").FirstChild("ZAP
P").FirstChild("VBS_RUNTIME_PARAMS").FirstChild("Broadcast").ToElement();
        TiXmlElement*  Broadcast_Pid  =  new  TiXmlElement("BROADCAST_PID");
        TiXmlText  *text  =new  TiXmlText("7215");
        Broadcast_Pid->SetAttribute("info","the  pid");
        Broadcast_Pid->LinkEndChild(text);
        Broadcast->LinkEndChild(Broadcast_Pid);
```

将在 SOURCE_3 后添加新的节点:

```
<BROADCAST_PID  info="the  pid">7215</BROADCAST_PID>
```

四,最后说一下中文乱码的问题

乱码是由于 GB2312 与 UTF8 之间转换不当造成的,tinyxml 在处理 UTF8 本身没有问题,当你打开一个 UTF8 的文档,可以在加载的时候指定 UTF8 的方式,或者文档声明处指明的编码格式,tinyxml 会按照相应的编码格式加载,但很多时候当我们输出或写入中文字段时会出现乱码,无论在内存,

还是打印出来的内容.这是因为我们的软件通常是GB2312编码,而读取或写入的内容是UTF8,自然就会出错.可以借助网上的两个函数来实现转换(原作者不详):

```
        void  ConvertUtf8ToGBK(CString&  strUtf8)
        {
                int  len=MultiByteToWideChar(CP_UTF8,  0,  (LPCTSTR)strUtf8,  -1,  NULL,0);
                unsigned  short  *  wszGBK  =  new  unsigned  short[len+1];
                memset(wszGBK,  0,  len  *  2  +  2);
                MultiByteToWideChar(CP_UTF8,  0,  (LPCTSTR)strUtf8,  -1,  wszGBK,  len);

                len  =  WideCharToMultiByte(CP_ACP,  0,  wszGBK,  -1,  NULL,  0,  NULL,  NULL);
                char  *szGBK=new  char[len  +  1];
                memset(szGBK,  0,  len  +  1);
                WideCharToMultiByte  (CP_ACP,  0,  wszGBK,  -1,  szGBK,  len,  NULL,NULL);

                strUtf8  =  szGBK;
                delete[]  szGBK;
                delete[]  wszGBK;
        }



        void  ConvertGBKToUtf8(CString&  strGBK)
        {
                int  len=MultiByteToWideChar(CP_ACP,  0,  (LPCTSTR)strGBK,  -1,  NULL,0);
                unsigned  short  *  wszUtf8  =  new  unsigned  short[len+1];
```

```
                memset(wszUtf8,  0,  len  *  2  +  2);
                MultiByteToWideChar(CP_ACP,  0,  (LPCTSTR)strGBK,  -1,  wszUtf
8,  len);

                len  =  WideCharToMultiByte(CP_UTF8,  0,  wszUtf8,  -1,  NUL
L,  0,  NULL,  NULL);
                char  *szUtf8=new  char[len  +  1];
                memset(szUtf8,  0,  len  +  1);
                WideCharToMultiByte  (CP_UTF8,  0,  wszUtf8,  -1,  szUtf8,  le
n,  NULL,NULL);

                strGBK  =  szUtf8;
                delete[]  szUtf8;
                delete[]  wszUtf8;
        }
```

当然,你也可以用 MultiByteToWideChar,WideCharToMultiByte 函数自己实现转换.以上是简单应用的几个举例,理解他们,相信你已经能写出满足自己需要的代码了.