[**交叉引用的解决方法——类声明的应用**](http://blog.csdn.net/luoweifu/article/details/46854535)

标签： [C++](http://www.csdn.net/tag/C%2b%2b)[交叉引用](http://www.csdn.net/tag/%e4%ba%a4%e5%8f%89%e5%bc%95%e7%94%a8)[类的声明](http://www.csdn.net/tag/%e7%b1%bb%e7%9a%84%e5%a3%b0%e6%98%8e)

2015-07-12 23:00 2207人阅读 [评论](http://blog.csdn.net/luoweifu/article/details/46854535#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/luoweifu/article/details/46854535#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

C++高级编程（6） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得用于任何商业用途，转载请注明出处。

目录[(?)[+]](http://blog.csdn.net/luoweifu/article/details/46854535)

交叉引用的解决方法

什么是交叉引用？

什么是交叉引用？一言以蔽之，就是：**A类中包含B类的对象，B类中包含A类的对象。**

以一场景为例

我们先来看一个场景。假设有一个电子文档(Document)、一个文档下有多个页(Page)，每个页下有多个文本单元(TextUnit,表示文档内元素的基本单位),一个文档中的所有文本单元对象都有唯一的ID。这样每创建一个文本单元时都要为其设置一个唯一的ID，我们在Document类中就需要一个生成唯一ID的方法为所有的文本单元创建唯一标识。于是我们就会有下面的类关系设计图：

   
图1 ：类的关系图

于是我们想当然的会有这样的代码：

**TextUnit.h：**

#pragma once

class TextUnit

{

public:

TextUnit(void);

TextUnit(int id);

~TextUnit(void);

public:

int GetId() { return m\_id; }

void SetId(int id) { m\_id = id; }

private:

int m\_id; //文本对象的唯一标识

};

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15

**TextUnit.cpp：**

#include "StdAfx.h"

#include "TextUnit.h"

TextUnit::TextUnit(void)

{

}

TextUnit::TextUnit( int id ) : m\_id(id)

{

}

TextUnit::~TextUnit(void)

{

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15

**Page.h：**

#pragma once

#include <vector>

#include "TextUnit.h"

#include "Document.h"

typedef std::vector<TextUnit\*> VecTextUnit;

class Page

{

public:

Page(Document\* pDocument);

~Page(void);

public:

//添加一个文本单元

TextUnit\* AddTextUnit();

private:

Document\* m\_pDocument; //文档对象,用于获得ID

VecTextUnit\* m\_pvecTextUnits; //文本单元对象

};

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22

**Page.cpp：**

#include "StdAfx.h"

#include "Page.h"

Page::Page(Document\* pDocument) : m\_pDocument(pDocument), m\_pvecTextUnits(new VecTextUnit)

{

}

Page::~Page(void)

{

for (VecTextUnit::iterator iter = m\_pvecTextUnits->begin(); iter != m\_pvecTextUnits->end(); ++ iter )

{

delete[] \*iter;

\*iter = NULL;

}

m\_pvecTextUnits->clear();

delete[] m\_pvecTextUnits;

m\_pvecTextUnits = NULL;

}

TextUnit\* Page::AddTextUnit()

{

int id = m\_pDocument->GenerateId();

TextUnit\* pTextUnit = new TextUnit(id);

if (pTextUnit == NULL)

{

return NULL;

}

m\_pvecTextUnits->push\_back(pTextUnit);

return pTextUnit;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30

**Document.h：**

#pragma once

#include <vector>

#include "Page.h"

typedef std::vector<Page\*> VecPage;

class Document

{

public:

Document(void);

~Document(void);

public:

//生成本文档内唯一的文本对象ID

int GenerateId();

//添加一页

Page\* AddPage();

private:

static int s\_id; //用于生成唯一的ID

VecPage\* m\_pvecPages; //所有的页

};

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23

**Document.cpp：**

#include "StdAfx.h"

#include "Document.h"

int Document::s\_id = 0;

Document::Document(void) : m\_pvecPages(new VecPage)

{

}

Document::~Document(void)

{

for (VecPage::iterator iter = m\_pvecPages->begin(); iter != m\_pvecPages->end(); ++ iter )

{

delete[] \*iter;

\*iter = NULL;

}

m\_pvecPages->clear();

delete[] m\_pvecPages;

m\_pvecPages = NULL;

}

int Document::GenerateId()

{

return s\_id ++;

}

Page\* Document::AddPage()

{

Page\* pPage = new Page(this);

if (pPage == NULL)

{

return NULL;

}

m\_pvecPages->push\_back(pPage);

return pPage;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36

编译

好，代码写完了，我们对它进行编译，这时你会发现一堆的错误：

1>d:\博客文章\c++高级编辑\projects\crossreference\crossreference\document.h(6): error C2065: ‘Page’ : undeclared identifier   
1>d:\博客文章\c++高级编辑\projects\crossreference\crossreference\document.h(6): error C2059: syntax error : ‘>’   
1>d:\博客文章\c++高级编辑\projects\crossreference\crossreference\document.h(9): error C2143: syntax error : missing ‘;’ before ‘{’   
…

1>d:\博客文章\c++高级编辑\projects\crossreference\crossreference\page.h(12): error C2061: syntax error : identifier ‘Document’   
1>d:\博客文章\c++高级编辑\projects\crossreference\crossreference\page.h(20): error C2143: syntax error : missing ‘;’ before ‘\*’   
1>d:\博客文章\c++高级编辑\projects\crossreference\crossreference\page.h(20): error C4430: missing type specifier - int assumed. Note: C++ does not support default-int   
1>d:\博客文章\c++高级编辑\projects\crossreference\crossreference\page.h(20): error C4430: missing type specifier - int assumed. Note: C++ does not support default-int   
…

原因分析

这是因为   
1. C++中，在创建或使用一个类时，这个类必须被定义完整(就是一个完整的类型)；   
2. 类的定义也可以和函数一样分成两步，先声明后定义。

class T; //声明一个类

这种声明也被称为**前向声明**，在程序中引入名字T，并指明T是一种类类型。此时的T在它被定义之前是**不完整的类型**（incomplete type)，也就是说只知道它是一种类类型，并不知道它有哪些成员，但可定义这个类型(T)的指针和引用。

类的定义：

class T

{

// todo: 定义类的成员(属性和方法)

};

只有类(T)定义完成，它才是一个完整的类型，才是可见的（才可被创建和使用）。

而我们的程序现在就出现这种很有意思的状态：在定义Document时，发现Page还没有定义完整(Document中有Page类型的成员)；在定义Page的时候发现Document还没有定义完整(Page中有Document类型的指针对象)。也就是说Document不知道Page，Page不知道Docunent，这时就像两只狗打架，A狗咬着B狗的尾巴，B狗咬着A狗的尾巴。

解决方法

1.在Document.h文本中加入Page类的声明：calss Page; 把include “Page.h”放到Document.cpp中。   
2.Page.h文本中加入Page类的声明：calss Document; 把include “Document.h”放到Page.h中。

代码如下:   
**Document.h：**

#pragma once

#include <vector>

class Page;

typedef std::vector<Page\*> VecPage;

class Document

{

public:

Document(void);

~Document(void);

public:

//生成本文档内唯一的文本对象ID

int GenerateId();

//添加一页

Page\* AddPage();

private:

static int s\_id; //用于生成唯一的ID

VecPage\* m\_pvecPages; //所有的页

};

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24

**Document.cpp：**

#include "StdAfx.h"

#include "Document.h"

#include "Page.h"

int Document::s\_id = 0;

Document::Document(void) : m\_pvecPages(new VecPage)

{

}

Document::~Document(void)

{

for (VecPage::iterator iter = m\_pvecPages->begin(); iter != m\_pvecPages->end(); ++ iter )

{

delete[] \*iter;

\*iter = NULL;

}

m\_pvecPages->clear();

delete[] m\_pvecPages;

m\_pvecPages = NULL;

}

int Document::GenerateId()

{

return s\_id ++;

}

Page\* Document::AddPage()

{

Page\* pPage = new Page(this);

if (pPage == NULL)

{

return NULL;

}

m\_pvecPages->push\_back(pPage);

return pPage;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37

**Page.h：**

#pragma once

#include <vector>

#include "TextUnit.h"

class Document;

typedef std::vector<TextUnit\*> VecTextUnit;

class Page

{

public:

Page(Document\* pDocument);

~Page(void);

public:

//添加一个文本单元

TextUnit\* AddTextUnit();

private:

Document\* m\_pDocument; //文档对象,用于获得ID

VecTextUnit\* m\_pvecTextUnits; //文本单元对象

};

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23

**Page.cpp：**

#include "StdAfx.h"

#include "Page.h"

#include "Document.h"

Page::Page(Document\* pDocument) : m\_pDocument(pDocument), m\_pvecTextUnits(new VecTextUnit)

{

}

Page::~Page(void)

{

for (VecTextUnit::iterator iter = m\_pvecTextUnits->begin(); iter != m\_pvecTextUnits->end(); ++ iter )

{

delete[] \*iter;

\*iter = NULL;

}

m\_pvecTextUnits->clear();

delete[] m\_pvecTextUnits;

m\_pvecTextUnits = NULL;

}

TextUnit\* Page::AddTextUnit()

{

int id = m\_pDocument->GenerateId();

TextUnit\* pTextUnit = new TextUnit(id);

if (pTextUnit == NULL)

{

return NULL;

}

m\_pvecTextUnits->push\_back(pTextUnit);

return pTextUnit;

}