

昵称：[樱桃小锤子](#)  
园龄：[3年5个月](#)  
粉丝：[23](#)  
关注：[2](#)  
[+加关注](#)

<	2011年8月						>
日	一	二	三	四	五	六	
31	1	<a href="#">2</a>	<a href="#">3</a>	4	5	6	
7	<a href="#">8</a>	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	<a href="#">25</a>	26	27	
28	29	30	31	1	2	3	
4	5	6	7	8	9	10	

搜索

找找看

谷歌搜索

常用链接

[我的随笔](#)  
[我的评论](#)  
[我的参与](#)  
[最新评论](#)  
[我的标签](#)

我的标签

[Android](#)(6)  
[C++](#)(5)  
[Qt](#)(5)  
[科普](#)(3)  
[NDK](#)(2)  
[Python](#)(2)

随笔档案(11)

[2013年12月](#) (1)  
[2013年4月](#) (2)  
[2013年3月](#) (1)  
[2012年12月](#) (2)  
[2011年10月](#) (1)  
[2011年8月](#) (4)

最新评论

[博客园](#) [首页](#) [博问](#) [闪存](#) [新随笔](#) [联系](#) [订阅](#) [XML](#) [管理](#)

随笔-11 评论-59 文章-0 trackbacks-0

Qt那点事儿（三） 论父对象与子对象的关系

## 第三回 父与子

70后的道友都应该看过这么一部片子叫做<<父子情深>>。讲述的是一个小男孩患了绝症,父亲为了满足他的愿望,让已关门的游乐园为他们父子俩重新开放。在游乐园尽情地玩耍后,最后小孩子在父亲的怀中安详地闭上了眼睛。缓缓转动的摩天轮,配着淡淡忧伤的曲调,这一刻哥泪流满面。谁说世上只有妈妈好,父爱也顶半边天。此时台下的众多男道友热泪盈眶，不约而同地起立鼓掌。史上最大的冤屈，终于得以昭雪。

但是人世间这种真挚的父爱也存在于Qt中吗？ 对此，从小缺乏父爱的张无忌小朋友给出了自己的答案，

```
1  #include <QDebug>
2  #include <QThread>
3
4  class MyTestA : public QObject
5  {
6      Q_OBJECT
7  public:
8
9  };
10
11 class MyTestB : public QObject
12 {
13 public:
14     MyTestB(QObject *parent):QObject(parent)
15     {
16
17     }
18 };
19
20 extern MyTestB *g_pMyTestB;
21 extern MyTestA *g_pMyTestA;
22 class MyTestC : public QThread
23 {
24     Q_OBJECT
25 public:
26
```

1. Re:Qt那点事儿（一）  
整个求知的过程值得我们学习  
--Tony.Works

阅读排行榜

1. QwebKit使用心得(4944)  
2. Qt那点事儿（三） 论父对象与子对象的关系(4700)  
3. Qt那点事儿（一）(3204)  
4. Qt那点事儿（二）(2416)  
5. 傲娇Android二三事之天不长地不久的Bitmap.compress(1963)

评论排行榜

1. 傲娇Android二三事之天不长地不久的Bitmap.compress(10)  
2. Qt那点事儿（一）(9)  
3. Qt那点事儿（二）(8)  
4. 傲娇Android二三事之操蛋的开发日记(第一回)(8)  
5. NDK开发笔记(一) NDK的安装(7)

推荐排行榜

1. NDK开发笔记(一) NDK的安装(5)  
2. 傲娇Android二三事之诡诡异异的图片加载(5)  
3. Qt那点事儿（一）(4)  
4. 傲娇Android二三事之天不长地不久的Bitmap.compress(3)  
5. 傲娇Android二三事之操蛋的开发日记(第一回)(2)

```
27     MyTestC():QThread(NULL)
28     {
29     }
30
31     void run()
32     {
33         exec();
34     }
35 };
36 int main(int argc, char *argv[])
37 {
38     QApplication app(argc, argv);
39
40     MyTestA a;
41
42     MyTestB b(&a);
43
44     MyTestC c;
45     c.start();
46
47     a.moveToThread(&c);
48     if(a.thread() == b.thread() && a.thread() != app.thread())
49     {
50         qDebug()<< "Both parent and son have the same thread";
51     }
52
53     return app.exec();
54 }
```

从容地按下了F5之后，只见输出窗口妥妥地输出了"Both parent and son have the same thread".

在Qt中，当一个对象被移到另一个线程时，他的所有子对象也会一并转移到另外那个线程。

一人移民，全家无忧阿。在场的一些兼职移民中介的道友叹道，简直就是一个经典的家庭移民案例。不愧是家有一父，如有一宝啊。

紧接着只见张无忌，对此代码稍作了修改，

```
1  class MyTestA : public QObject
2  {
3      Q_OBJECT
4  public:
5      };
6
7  class MyTestB : public QObject
8  {
9  public:
10     MyTestB(QObject *parent):QObject(parent)
11     {
12
13     }
14 };
15
```

```

16 extern MyTestB *g_pMyTestB;
17 extern MyTestA *g_pMyTestA;
18 class MyTestC : public QThread
19 {
20     Q_OBJECT
21 public:
22
23     MyTestC():QThread(NULL)
24     {
25     }
26
27     void run()
28     {
29         g_pMyTestA->moveToThread(this);
30         exec();
31     }
32 };
33
34 MyTestB *g_pMyTestB = NULL;
35 MyTestA *g_pMyTestA = NULL;
36 int main(int argc, char *argv[])
37 {
38     QApplication app(argc, argv);
39
40     MyTestA a;
41     g_pMyTestA = &a;
42
43     MyTestB b(&a);
44
45     MyTestC c;
46     c.start();
47
48     return app.exec();
49 }

```

却见output窗口打出,

```

"QObject::moveToThread: Current thread (0x2ff944) is not the object's thread (0x357b20).
Cannot move to target thread (0x2ff944)"

```

在Qt中，如果要切换对象的线程，不能到了目标线程里再调用**moveToThread**,此举会导致切换线程失败。

众人皆称，移民要合法，偷渡要不得啊。

就在众人嗟叹时，年轻气盛的无忌小友，又刷刷的写下了以下代码，

```

1  #include <QThread>
2
3  class MyTestA : public QObject
4  {
5      Q_OBJECT
6  public:
7

```

```
8 };
9
10 class MyTestB : public QObject
11 {
12 public:
13     MyTestB(QObject *parent):QObject(parent)
14     {
15     }
16 };
17
18 extern MyTestB *g_pMyTestB;
19 extern MyTestA *g_pMyTestA;
20 class MyTestC : public QThread
21 {
22     Q_OBJECT
23 public:
24
25     MyTestC(QObject *parent):QThread(parent)
26     {
27     }
28 };
29
30
31 class MyTest : public QDialog
32 {
33     Q_OBJECT
34
35 public:
36     MyTest(QWidget *parent = 0, Qt::WFlags flags = 0);
37     ~MyTest();
38
39 protected slots:
40     void onClick();
41
42
43 private:
44     Ui::MyTestClass ui;
45 };
46 //////////////////////////////////////
47 MyTest::MyTest(QWidget *parent, Qt::WFlags flags)
48     : QDialog(parent, flags)
49 {
50     ui.setupUi(this);
51     //set the window to be the top window
52     <span style="color: #ff0000;">this->setWindowFlags(windowFlags()|Qt::WindowStaysOnTopHint);</span>
53 }
54
55 MyTest::~MyTest()
56 {
57
```

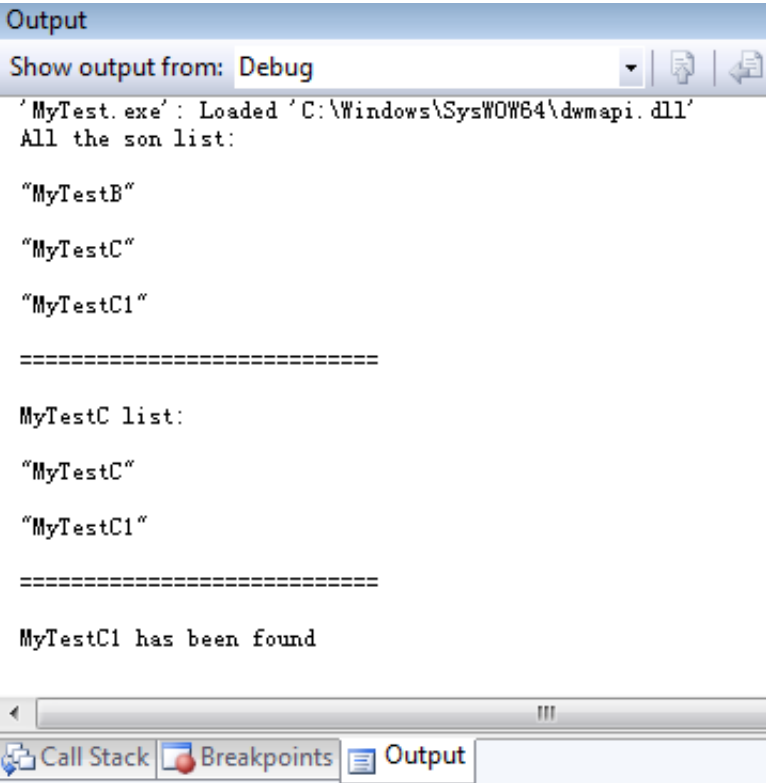
```

58     }
59
60 void MyTest::onClick()
61 {
62     <span style="color: #ff0000;">QMessageBox box(this);</span>
63     box.setText("i am at the top");
64     box.exec();
65 }
66
67 //////////////////////////////////////////////////main.cpp////////////////////////////////////
68 MyTestB *g_pMyTestB = NULL;
69 MyTestA *g_pMyTestA = NULL;
70 int main(int argc, char *argv[])
71 {
72     QApplication app(argc, argv);
73
74
75
76     MyTestA a;
77
78     <span style="color: #ff0000;">MyTestB *pB = new MyTestB(&a);</span>
79     <span style="color: #ff0000;">pB->setObjectName("MyTestB");</span>
80
81     <span style="color: #ff0000;">MyTestC *pC = new MyTestC(&a);</span>
82     <span style="color: #ff0000;">pC->setObjectName("MyTestC");</span>
83
84     <span style="color: #ff0000;">pC = new MyTestC(&a);</span>
85     <span style="color: #ff0000;">pC->setObjectName("MyTestC1");</span>
86
87     <span style="color: #ff0000;">QList<QObject*> list = a.findChildren<QObject*>();</span>
88     QList<QObject*>::iterator it;
89     qDebug()<<"All the son list: "<<"\r\n";
90     for(it = list.begin(); it != list.end() ; it++)
91     {
92         qDebug()<<(*it)->objectName()<<"\r\n";
93     }
94     qDebug()<<"===== "<<"\r\n";
95
96
97     <span style="color: #ff0000;">QList<MyTestC*> listC = a.findChildren<MyTestC*>();</span>
98     QList<MyTestC*>::iterator itC;
99     qDebug()<<"MyTestC list: "<<"\r\n";
100    for(itC = listC.begin(); itC != listC.end() ; itC++)
101    {
102        qDebug()<<(*itC)->objectName()<<"\r\n";
103    }
104    qDebug()<<"===== "<<"\r\n";
105
106    <span style="color: #ff0000;">MyTestC *pC1 = a.findChild<MyTestC*>("MyTestC1");</span>
107    if(pC1)

```

```
108     {
109         qDebug()<<"MyTestC1 has been found"<<"\r\n";
110     }
111
112     MyTest win;
113     win.show();
114
115     return app.exec();
116 }
```

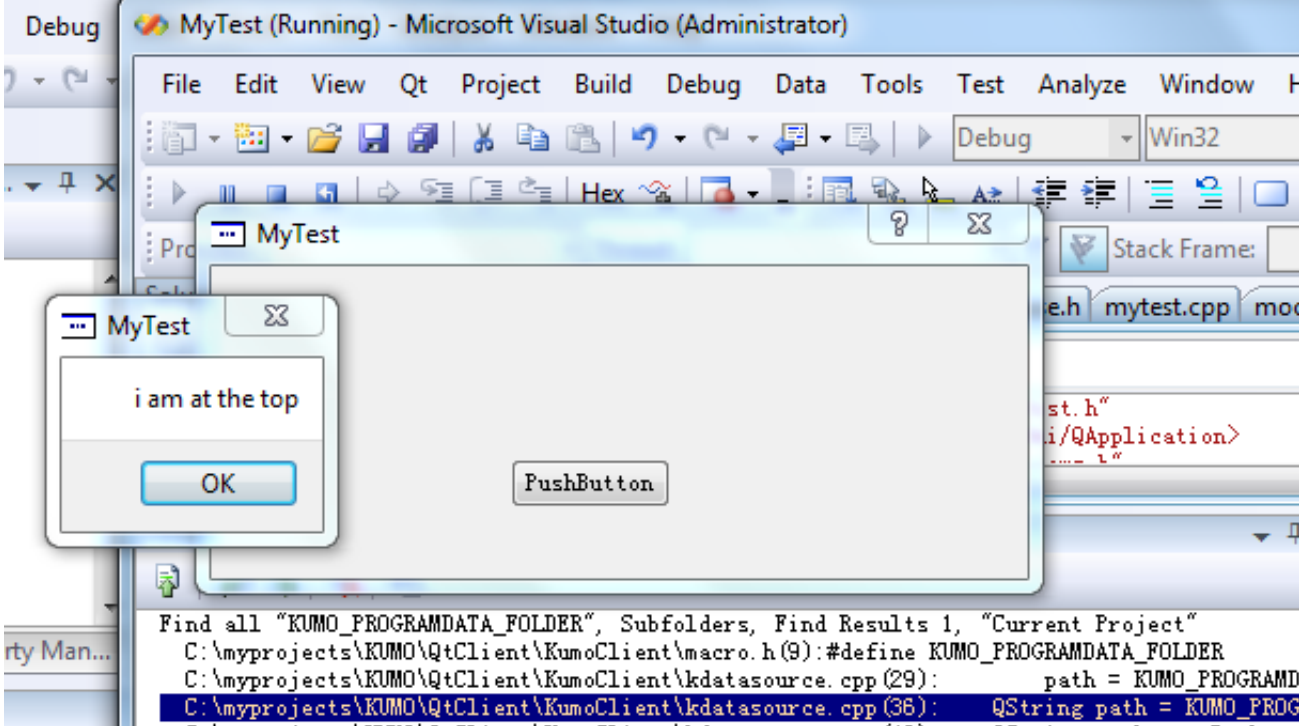
然后销魂的转身一点，只见



在Qt中，我们可以通过**findChild,findChildren,qFindChild,qFindChildren**,来遍历所有的子对象，同时我们可以通过指定类型，来得到所有的指定类型的子对象，当然也可以通过对象名字来索引。比如**m\_dlg.findChildren<QPushButton\*>()**;通过这个函数我们可以轻松的遍历出对话框中所有的**QPushButton**子对象，这样对我们诸如换语言的操作提供了便利。换句话说，Qt的父对象也起到了一个容器的作用，我们有时可以利用这一点，把父对象作为一个容器处理。

众人不禁赞道，知子莫如父啊。

无忌小友看在眼里，喜在心头。只见他又继续点击F5,弹出了一个对话框，



此对话框设置了Top属性，使之能够在所有其它应用程序窗口之上(this->setWindowFlags(windowFlags()|Qt::WindowStaysOnTopHint);)。然后又点击了PushButton，弹出了一子对话框。只见子对话框也自动继承了父窗口的属性，成为了Top window。

在Qt中，我们只需在父窗口设置某些属性（比如**Top, bottom**），子窗口将自动获得这些属性，使开发者不用为了保持子窗口与父窗口的一致性，每个窗口一个一个去设置。提高了开发效率。

众人皆叹，有父如此，子欲何求。老子干活，儿子享福啊。此时一股浓浓的父爱弥漫在武当大殿中。谁说父爱不顶半边天？此时的男道友們心潮澎湃，激动之余不禁拨通了"流言终结者"的制作组电话。

而反观另外一些道友，眼看她们引以为傲的优势，将被击得荡然无存。她们不甘心失败，一遍遍的看着代码，企图找出一丝破绽来。终于，一位女道友面带冷笑，指着代码说道，“无忌道友，此程序好似有内存泄露，不知对否”。众人心头一紧，Qt往日的无耻又浮现在了人们心头。

但见无忌小友手持羽扇，迎风而立，露出招牌般的正太式微笑，徐徐说道，“早知道友会有此一问。”接着从怀中取出一本写有“九阳真经”的古籍，翻了开来。只见一幅制作精美具有扶桑画风的彩图映入了众人的眼帘，图下面写着“伴我成长的女人们”。张无忌脸色一红，尴尬地咳嗽了一声，又继续翻到了下一页，只见上面写着，

```
1  QObject::~~QObject()
2  {
3      . . . . .
4
5      if (!d->children.isEmpty())
6          d->deleteChildren();
7
8      . . . . .
9  }
10
11 void QObjectPrivate::deleteChildren()
12 {
13     const bool reallyWasDeleted = wasDeleted;
14     wasDeleted = true;
15     // delete children objects
16     // don't use qDeleteAll as the destructor of the child might
17     // delete siblings
18     for (int i = 0; i < children.count(); ++i) {
19         currentChildBeingDeleted = children.at(i);
20         children[i] = 0;
21         delete currentChildBeingDeleted;
22     }
23     children.clear();
```

```
24     currentChildBeingDeleted = 0;
25     wasDeleted = reallyWasDeleted;
26 }
27
```

在Qt中，当以**QObject**为父类的对象析构时，他会自动删除它所包含的所有子对象，实现了简单的垃圾回收机制，避免了内存泄露。所以开发时可以考虑，每个**new**出来的对象尽量设置父对象，这样即使未显示调用**delete**，只要保证父对象被析构，就能避免内存泄露。

武当大殿沸腾了，观众们被Qt父子情深般的精彩表演深深震撼了。”学Qt，得永生”的口号响彻云霄（春哥泪流满面）。《流言终结者》主持人杰米和亚当宣布，人类史上最大的流言"父子不如X子亲"终结了。节目赞助商Intel鉴于此期节目在CCAV上99.99%的收视率，以4.44亿RMB天价强行插入了一条广告“Intel，给Qt一颗奔腾的芯”。

而Qt的代言人无忌小友，获得了道教界一年一度以道家镇教之宝命名的，最高荣誉“八卦”奖。当从道教最高精神领袖“张三丰”手中接过雕有“冠希”前辈手拿camera的小金像，正要发表获奖感言的时候，一道剑光闪过。

正所谓伯乐不常有，但搅屎棍却常在。

只见人见人怕，鬼见鬼愁，考试只给59分的灭绝师太，手握倚天剑，刷刷的修改了张无忌的代码。

```
1  class MyTestA : public QObject
2  {
3      Q_OBJECT
4  public:
5  };
6
7  class MyTestB : public QObject
8  {
9  public:
10     MyTestB(QObject *parent):QObject(parent)
11     {
12
13     }
14 };
15
16 extern MyTestB *g_pMyTestB;
17 extern MyTestA *g_pMyTestA;
18 class MyTestC : public QThread
19 {
20     Q_OBJECT
21 public:
22
23     MyTestC():QThread(NULL)
24     {
25     }
26
27     void run()
28     {
29         exec();
30     }
31 };
32
33
34 MyTestB *g_pMyTestB = NULL;
35 MyTestA *g_pMyTestA = NULL;
```



```
36   int main(int argc, char *argv[])
37   {
38       QApplication app(argc, argv);
39
40       MyTestA a;
41
42       MyTestB b(&a);
43
44       MyTestC c;
45       c.start();
46
47       b.moveToThread(&c);
48
49       return app.exec();
50   }
```

执行此段代码后，众人皆惊。只见output窗口输出了“QObject::moveToThread: Cannot move objects with a parent”。

灭绝师太斜眼冷笑道：“黄口小儿，安能善言人伦乎？”

由此可见**Qt**中，子对象不能脱离父对象，单独切换到与父对象不同的线程中。

此时的张无忌面色惨白。但灭绝师太誓将张无忌搞臭到底，以不负灭绝的美名。只见她又修改了一段代码，

```
class MyTestA : public QObject
{
    Q_OBJECT
public:
};

class MyTestB : public QObject
{
public:
    MyTestB(QObject *parent):QObject(parent)
    {

    }
};

extern MyTestB *g_pMyTestB;
extern MyTestA *g_pMyTestA;
class MyTestC : public QThread
{
    Q_OBJECT
public:

    MyTestC():QThread(NULL)
    {

    }

    void run()
    {
```

```
        exec();
    }
};

MyTestB *g_pMyTestB = NULL;
MyTestA *g_pMyTestA = NULL;
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    MyTestA *pA = new MyTestA;

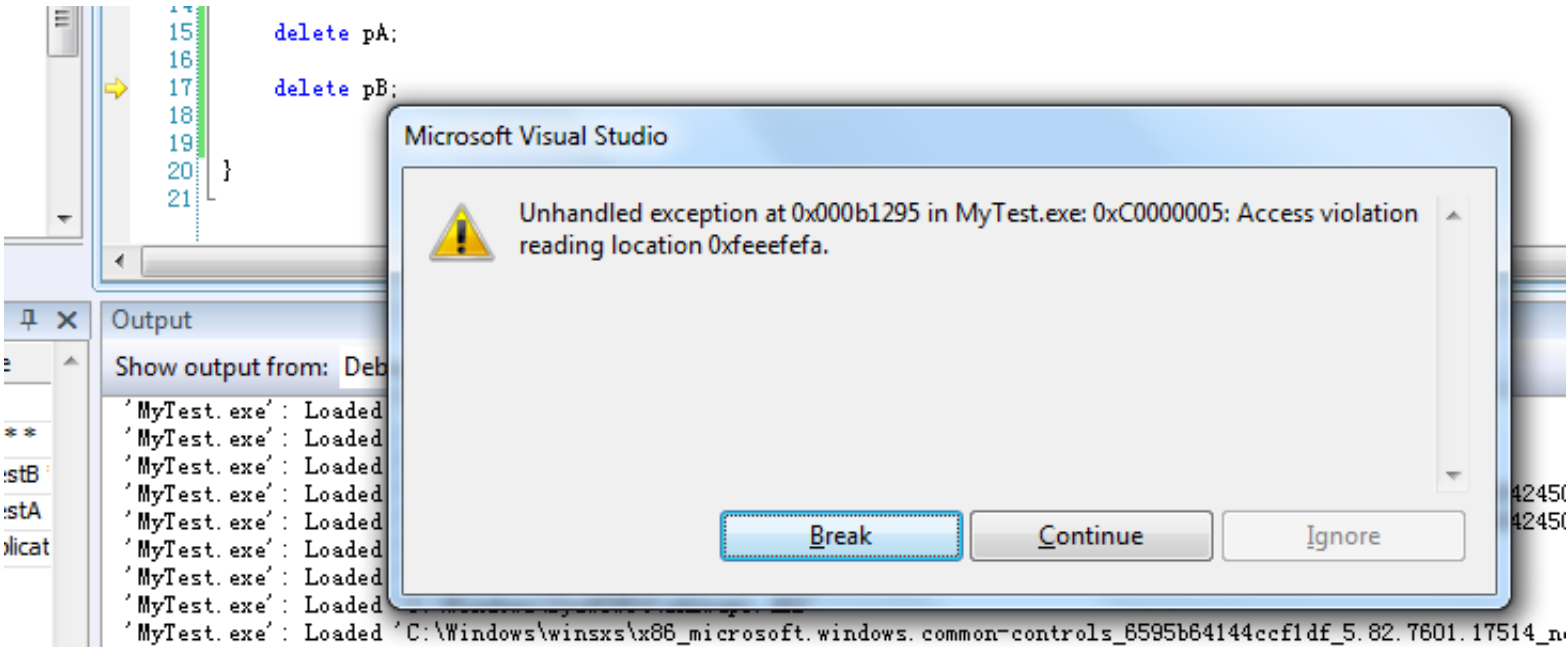
    MyTestB *pB = new MyTestB(pA);

    delete pA;

    delete pB;

}
```

只见程序蹦出了警告对话框，



程序直接崩溃了。与之同时崩溃的，还有众男道友的心。

而张无忌啪跌坐在地上，万念俱灰。与霆锋哥相拥痛哭，为什么上一辈的悲剧，又在我们身上重演。

对于Qt子对象而言，不能在父对象删除后，再删除自己。因为父对象析构时，会删除所有的子对象，此时子对象再删除，会引起二次析构。

所以如果子对象要切换到另一个线程或者避免被父对象删除，则需要调用**setParent(NULL)**,解除父子关系。

灭绝师太仰天长笑道“Qt名为父子，实乃黑帮。”

太史公评曰：“一入Q门深似海，从此萧郎是路人”。

瑟瑟风中，只见张无忌将自己多年的呕心力作<<我与Qt之间不得不说的故事>>付之一炬，飘然而去。从此之后，弃码从武，苦练九阳真经，终成一代大侠，名满江湖，这当然都是后话。

欲知后事如何，请听下回分解。

标签: C++, Qt


绿色通道:


好文要顶

关注我

收藏该文

与我联系





樱桃小锤子  
关注 - 2  
粉丝 - 23  
[+加关注](#)

0

0

(请您对文章做出评价)

« 上一篇: [Qt那点事儿（二）](#)

» 下一篇: [为什么新安装的pywin32不能用](#)

posted on 2011-08-25 10:05 [樱桃小锤子](#) 阅读(4700) 评论(1) [编辑](#) [收藏](#)

评论:

#1楼 2011-09-21 20:33 | [散客游](#)  
<<我与QT二三事>>

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#)网站首页。

- 【[免费课程](#)】案例：图片展示特效
- 【[推荐](#)】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- [融云](#)，免费为你的App加入IM功能——让你的App“聊”起来！！



最新**IT**新闻:

- [刷大墙/送财神/送春联 网贷平台进军农村也是拼了!](#)
- [阿里与工商总局掐架 受伤的是美国股民](#)
- [为啥手机厂商一窝蜂地烧Hi-Fi，做耳机?](#)
- [你在淘宝上的买买买，能让阿里巴巴给你的信用打几分呢?](#)
- [亚马逊将分拆AWS云计算服务](#)
- » [更多新闻...](#)



0基础3个月学会Android开发

挑战月薪1W+

最新知识库文章:

- [大数据架构和模式（五）——对大数据问题应用解决方案模式并选择实现它的产品](#)
- [大数据架构和模式（四）——了解用于大数据解决方案的原子模式和复合模式](#)
- [大数据架构和模式（三）——理解大数据解决方案的架构层](#)
- [大数据架构和模式（二）——如何知道一个大数据解决方案是否适合您的组织](#)
- [大数据架构和模式（一）——大数据分类和架构简介](#)
- » [更多知识库文章...](#)