

个人资料



牛搞



访问：1048764次

积分：11197

等级：BLOG > 7

排名：第427名

原创：55篇 转载：8篇

译文：76篇 评论：393条

文章搜索

博客专栏



android sdk

文章：79篇

阅读：647655



流媒体开发

文章：30篇

阅读：288622

文章分类

[HTML 5全掌控](#) [那些年我们追过的Wrox精品红皮计算机图书](#) [PMBOK第五版精讲视频教程](#) [CSDN JOB带你坐飞机回家过年](#)

[置顶] Qt属性系统详解

分类：c++ Qt4

2012-11-26 11:54

4396人阅读

评论(0)

收藏

举报

目录(?)

[+]

Qt提供了一个绝妙的属性系统。跟那些由编译器提供的属性差不多。然而，作为一个独立于编译特性的库，Qt不依赖于非标准的编译特性，比如__property 或[property]。Qt可以在任何平台上的标准编译器下编译。Qt属性系统基于元数据对象系统——就是那个提供了对象内置信号和槽通讯机制的家伙。

声明属性需要什么

要声明一个属性，需在继承自QObject的类中使用Q_PROPERTY()宏。

Q_PROPERTY(type name

READ getFunction

[WRITE setFunction]

[RESET resetFunction]

[NOTIFY notifySignal]

[DESIGNABLE bool]

[SCRIPTABLE bool]

[STORED bool]

[USER bool]

[CONSTANT]

[FINAL])

下面是一些典型的声明属性的示例：

[cpp]

```
01. Q_PROPERTY(bool focus READ hasFocus)
02. Q_PROPERTY(bool enabled READ isEnabled WRITE setEnabled)
03. Q_PROPERTY(QCursor cursor READ cursor WRITE setCursor RESET unsetCursor)
```

[android开发](#) (80)

[流媒体](#) (26)

[python](#) (2)

[Qt4](#) (6)

[c++](#) (8)

[开发工具](#) (4)

[.net](#) (1)

[android开发 jni](#) (0)

[jni](#) (1)

[OpenGL ES](#) (1)

[项目管理](#) (1)

[winphone](#) (2)

[外行看开发](#) (1)

[web](#) (1)

[看热闹](#) (0)

文章存档

[2014年12月](#) (1)

[2014年06月](#) (1)

[2013年11月](#) (1)

[2013年08月](#) (1)

[2013年06月](#) (1)

展开

阅读排行

[Android Service 详解四：](#) (39294)

[android Fragments详解I](#) (37860)

[android Fragments详解I](#) (35185)

[Android NDK开发轻松入](#) (25637)

[live555学习笔记1－引子](#) (22027)

[live555学习笔记2－基础](#) (19376)

[Android开发：什么是IBi](#) (18387)

[live555学习笔记3－消息](#) (18252)

[Android Service 详解二：](#) (16957)

[live555学习笔记7－RTP](#) (16780)

评论排行

[终于完成RTP/RTSP代理](#) (25)

[ffmpeg源码分析之媒体打](#) (14)

- 一个属性的行为就像类的数据成员，但是它还具有附加的特性，这些特性可以被元数据对象系统操作。这些特性是：
需要一个READ访问器函数。用于读属性的值。理想情况下，有一个不变的函数用于此目的，并且它必须返回属性的类型的值或指针或引用。例如，QWidget::focus是一个只读的属性，它对应一个读函数：
QWidget::hasFocus()。
- 一个可选的WRITE访问器函数。它用于设置属性的值。它必须返回空并且至少具有一个参数，参数是属性类型的值或指针或引用。例如：QWidget::enabled具有WRITE函数QWidget::setEnabled()。只读属性不需要写函数。例如，QWidget::focus没有对应的写函数。
- 一个可选的RESET函数。用于设置属性的值到它的默认值。例如：QWidget::cursor具有典型的READ和WRITE函数，QWidget::cursor()和QWidget::setCursor()，并且它也具有一个RESET函数，QWidget::unsetCursor()。RESET函数必须返回void并且不带有任何参数。
- 一个可选的NOTIFY信号。如果被定义了，信号将在属性的值改变时发出。信号必须带有一个参数，这个参数的类型必须与属性相同；参数保存的是属性的新值。
- 一个DESIGNABLE变量表明此属性是否在界面设计器的属性编辑器中出现。大多数属性是可见的，除了为这个变量传入true或false，你还可以指定一个bool型的成员函数。
- SCRIPTABLE变量表明这个属性是否可以被一个脚本引擎操作（默认是true）。你也可以赋予它true或false或bool型函数。
- STORED变量表明了属性是否被认为是独立存在还是依赖于其它的值而存在。它也表明是否在保存对象状态时保存此属性的值。大多数属性都是需要保存的，但是，如QWidget::minimumWidth()就是不被保存的，因为它的值是从另一个属性QWidget::minimumSize()得来的。
- USER变量表明属性是否被设计为面向用户的或用户可修改的类属性。通常，每个类只有一个USER属性。例如，QAbstractButton::checked是按钮类的用户可修改属性。注意QItemDelegate获取和设置widget的USER属性。
- CONSTANT的出现表明属性的值是不变的。对于一个object实例，常量属性的READ方法在每次被调用时必须返回相同的值。此常量值可能在不同的object实例中不相同。一个常量属性不能具有WRITE方法或NOYIFY信号。
- FINAL变量的出现表明属性不能被派生类所重写。有些情况下，这可以用于效率优化，但不是被moc强制的。程序员必须永远注意不能重写一个FINAL属性。

READ，WRITE和RESET函数都可以被继承。它们也可以是虚函数。当它们在被多重继承中被继承时，它们必须出现在第一个被继承的类中。

属性的类型可以是被QVariant支持的所有类型，也可以是用户定义的类型。在下面的例子中，类QDate被当作用户自定义类型。

Q_PROPERTY(QDate data READ getDate WRITE setDate)

因为QDate是用户定义的，你必须包含<QDate>头文件。

对于QMap, QList和QValueList属性，属性的值是一个QVariant，它包含整个list或map。注意Q_PROPERTY字符串不能包含逗号，因为逗号会划分宏的参数。因此，你必须使用QMap作为属性的类型而不是QMap<QString, QVariant>。为了保持一致性，也需要用QList和QValueList而不是QList<QVariant>和

- 一步一步学android Oper (13)
- Live555学习笔记14—liv (12)
- android Fragments详解三 (12)
- android Fragments详解I (11)
- Android NDK开发轻松入 (10)
- Android Service 详解二: (10)
- live555学习笔记2—基础 (10)
- live555学习笔记17—H2 (9)

推荐文章

- * 浅析总结 Java 内部类的一些使用与梳理
- * Qt for iOS, Qt 与Objective C混合编程
- * 教你写Android ImageLoader框架之基本架构
- * 三大运营商的游戏“刷金”漏洞解决方案
- * 百度地图开发（二）之添加覆盖物 上 地理信息和后地理信息

最新评论

- android进程与线程详解一:进程繁华未至: 挺好的，有个问题想要请教下，最近项目里面有拍照的功能（有一款手机会出现这样的问题），调用系统相机，启...
- 新版live555的问题SmartSmall: N天以契而不舍的苦逼型精神进行无数次折腾~~~~~
- 最新版ffmpeg源码分析二:transc liuxuejin: 能告诉这是那个版本么？
- android Fragments详解三:实现F xiongmaozhijin: 学习。
- Android Service 详解二：创建一 sysou: 放你吗的狗屁
- live555学习笔记2—基础类principl: 记得在学习计算机网络的时候好像有提到过：多播和广播仅用于UDP，所以live555中的groupso...
- RTSP 播放器 demo szlkbjb: 就是太贵
- 一步一步学android OpenGL ES. HaijunZhu: 楼主你好，我做基于webrtc的android应用，想加入哪个群但是，群已满，可否有其他的群让我加入...
- android activity详解二：Activity biandan1231: 楼主总结得很好，但我还是有一个疑问，为什么要必须首先调用父类的同一方

QValueList<QVariant>。

通过元数据对象系统读写属性

一个属性可以使用常规函数QObject::property()和QObject::setProperty()进行读写，不用知道属性所在类的任何细节，除了属性的名字。在下面的小代码片段中，调用QAbstractButton::setDown()和QObject::setProperty()都把属性设置为“down”。

```
[cpp]

01. QPushButton *button = new QPushButton;
02. QObject *object = button;
03. button->setDown(true);
04. object->setProperty("down", true);
```

通过WRITE操作器来操作一个属性是上面两者中更好的，因为它快并且在编译时给予更好的诊断帮助，但是以这种方式设置属性要求你必须在编译时了解其类。通过名字来操作属性使你可以操作在编译器你不了解的类。你可以在运行时发现一个类的属性们，通过查询它的QObject,QMetaObject和QMetaProerties。

```
[cpp]

01. QObject *object = ...
02. const QMetaObject *metaobject = object->metaObject();
03. int count = metaobject->propertyCount();
04. for (int i=0; i<count; ++i) {
05.     QMetaProperty metaproperty = metaobject->property(i);
06.     const char *name = metaproperty.name();
07.     QVariant value = object->property(name);
08.     ...
09. }
```

在上面的代码片段中，QMetaObject::property()被用于获取未知类中的属性的metadata。从metadata中获取属性名然后传给QObject::property()来获取

一个简单例子

假设我们有一个类MyClass，它从QObject派生并且在它的private区使用 了Q_OBJECT宏。我们想在MyClass类中声明一个属性来持续追踪一个Priority值。属性的值叫做priority，并且它的类型是一个在类MyClass中定义的叫作Priority的枚举。

我们在类的private区使用Q_PROPERTY()来声明属性。READ函数叫做priority,并且我们包含一个WRITE函数叫做setPriority。枚举类型必须使用Q_ENUMS()注册到元数据对象系统中。注册一个枚举类型使得枚举的名字可以在调用QObject::setProperty()时使用。我们还必须为READ和WRITE函数提供我们自己的声明。MyClass的声明看起来

法?? 虽然很久的文章了, , , ,

live555学习笔记10—h264 RTP

daminglanyu: 非常给力, 分析的非常透彻, 哥, 请教一个问题, H264VideoStreamParser::parse...

应该是这样的:

```
[cpp]

01. class MyClass : public QObject
02. {
03.     Q_OBJECT
04.     Q_PROPERTY(Priority priority READ priority WRITE setPriority)
05.     Q_ENUMS(Priority)
06. public:
07.     MyClass(QObject *parent = 0);
08.     ~MyClass();
09.     enum Priority { High, Low, VeryHigh, VeryLow };
10.     void setPriority(Priority priority);
11.     Priority priority() const;
12. };
```

READ函数是const的并且返回属性的类型。WRITE函数返回void并且具有一个属性类型的参数。元数据对象编译器强制做这些事情。

在有了一个指向MyClass实例的指针时, 我们有两种方法来设置priority属性:

```
[cpp]

01. MyClass *myinstance = new MyClass;
02. QObject *object = myinstance;
03. myinstance->setPriority(MyClass::VeryHigh);
04. object->setProperty("priority", "VeryHigh");
```

在此例子中, 枚举类型在MyClass中声明并被使用Q_ENUMS()注册到元数据对象系统中。这使得枚举值可以在调用setProperty()时做为字符串使用。如果枚举类型是在其它类中声明的, 那么我们就需要用枚举的全名 (如OtherClass::Priority), 并且这个其它类也必须从QObject中派生并且也要注册枚举类型。

另一个简单的Q_FLAGS()也是可用的。就像Q_ENUMS(), 它注册一个枚举类型, 但是它把枚举类型作为一个flag的集合, 也就是, 值可以用OR操作来合并。一个I/O类可能具有枚举值Read和Write并且QObject::setProperty()可以接受 Read|Write。此时应使用Q_FLAGS()来注册枚举值。

动态属性

QObject::setProperty()也可以用来在运行时向一个类的实例添加新的属性。当使用一个名字和值调用它时, 如果一个对应的属性已经存在, 并且如果值的类型与属性的类型兼容, 那么值就被存储到属性中, 然后返回true。如果值类型不兼容, 属性的值就不会发生改变, 就会返回false。但是如果对应名字的属性不存在, 那么一个新的属性就诞生了, 以传入的名字为名, 以传入的值为值, 但是依然会返回false。这表示返回值不能用于确定一个属性是否被设置值, 除非你已经知道这个属性已经存在于QObject中了。

注意动态属性被添加到单个实现的基础中, 也就是, 被添加到QObject, 而不是QMetaObject。一个属性可以从一个实例中删除, 通过传入属性的名字和非法的QVariant值给QObject::setProperty()。默认的QVariant构造器构造一个非法的QVariant。

动态属性可用QObject::property()来查询, 就行使用Q_PROPERTY()声明的属性一样。

属性和自定义类型

被属性使用的自定义类型需要使用Q_DECLARE_METATYPE()宏注册，以使它们的值能被保存在QVariant对象中。这使得它们可以用于被Q_PROPERTY()声明的静态类型中，也可以被用于动态类型中。

上一篇 Qt Tooltip详解

下一篇 Qt 动画详解一

顶
2

踩
1

主题推荐

界面设计 编译器 程序员 编辑器 实例

猜你在找

宏Q_OBJECT	Qt动画效果的幕后英雄QTimeLine
设置QPushButton的背景图片	Qt学习布局管理器QLayout类
QT QWidget设置窗体透明度方法汇总	如何让 Qt 的程序使用 Sleep
Qt入门－QLineEditsetInputMask	创建有个性的对话框之MFC篇一
Qt好书推荐	Qt5-Eclipse 与 在Qt creator中文输入

准备好了么？跳吧!

更多职位尽在 CSDN JOB

系统软件工程师	我要跳槽	ERP系统开发工程师	我要跳槽
先进科技（中国）有限公司	5-15K/月	CVTE	8-30K/月
安卓系统工程师	我要跳槽	大数据系统高级研发工程师	我要跳槽
CVTE	7-20K/月	北京好赞移动科技有限公司	15-25K/月



查看评论

暂无评论

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

核心技术类目

[全部主题](#) [Hadoop](#) [AWS](#) [移动游戏](#) [Java](#) [Android](#) [iOS](#) [Swift](#) [智能硬件](#) [Docker](#) [OpenStack](#)
[VPN](#) [Spark](#) [ERP](#) [IE10](#) [Eclipse](#) [CRM](#) [JavaScript](#) [数据库](#) [Ubuntu](#) [NFC](#) [WAP](#) [jQuery](#)
[BI](#) [HTML5](#) [Spring](#) [Apache](#) [.NET](#) [API](#) [HTML](#) [SDK](#) [IIS](#) [Fedora](#) [XML](#) [LBS](#) [Unity](#)
[Splashtop](#) [UML](#) [components](#) [Windows Mobile](#) [Rails](#) [QEMU](#) [KDE](#) [Cassandra](#) [CloudStack](#)
[FTC](#) [coremail](#) [OPhone](#) [CouchBase](#) [云计算](#) [iOS6](#) [Rackspace](#) [Web App](#) [SpringSide](#) [Maemo](#)
[Compuware](#) [大数据](#) [aptech](#) [Perl](#) [Tornado](#) [Ruby](#) [Hibernate](#) [ThinkPHP](#) [HBase](#) [Pure](#) [Solr](#)
[Angular](#) [Cloud Foundry](#) [Redis](#) [Scala](#) [Django](#) [Bootstrap](#)

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [银行汇款帐号](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) [webmaster@csdn.net](#) 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 070598 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved 