lpmygod的专栏

个人资料

lpmygod

访问： 14165次
积分： 329
等级： BLOG 2
排名： 千里之外

原创： 15篇 转载： 26篇
译文： 0篇 评论： 4条

文章分类

HACK (7)
光通信 (1)
生活 (10)
C++ (3)
QT (14)
随笔 (1)

文章存档

2014年11月 (3)
2014年10月 (5)
2014年09月 (8)
2014年08月 (2)
2013年11月 (5)

展开

阅读排行

FB面试（转自Roba Blog (1824)
灰鸽子配置使用手册 (1552)
【转】一个黑客所需的基 (1068)
如何使自己变得优雅 (1064)
栈溢出攻击 (1057)
如何学习黑客技术 (787)
光接收机 (754)
男人读懂女人心必须知道 (356)
rep+stos+dword+ptr+[ec (338)

Qt中事件处理的方法和实例

分类： QT 2014-09-08 11:45 210人阅读 评论(0) 收藏 举报

目录(?) [+]

一、Qt中事件处理的方式

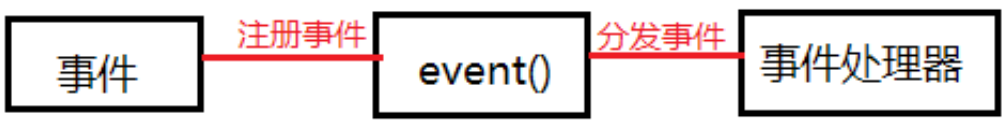
1、事件处理模式一

首先是事件源产生事件，最后是事件处理器对这些事件进行处理。然而也许大家会问，

Qt中有这么多类的事件，我们怎么样比较简便的处理每个事件呢？设想，如果是每个事件都对应同一个事件处理器，在该事件处理器中对不同的事件进行分类处理，这样的弊端有两点：第一，导致该事件处理器过于臃肿复杂；第二，这样不便于扩展，当系统新增加事件类型或者是我们需要使用到自定义事件时，就不得不修改Qt的源码来达到目的。所以Qt设计者的做法是针对不同类型的事件提供不同的事件处理器与之对应。这里又有一个问题了，Qt中是怎么让不同类型事件与之对应的事件处理器相关联的呢？我们不难猜想在事件和事件处理器中间必定有一个桥梁。这个桥梁就是QObject::event()函数，该函数是虚函数，QObject的子类例如QWidget都实现了该函数。该函数的主要功能是进行事件的分发，也就是将不同类型的事件与之相对应的事件处理器相关联，该函数并不对事件进行处理，真正的事件处理是在事件处理器中进行的。

例如：当QWidget产生QPaintEvent事件后，QWidget的event函数会将该事件分发给QWidget::paintEvent()事件处理器，这样该事件就得到处理了。

以上内容用一个图形表示就是：



2、事件处理模式二

很多时候，我们只对某些特定的事件比较关心，例如：鼠标单击或者键盘按下等事件。其它的事件我们并不关心它是否发生，也无需对它们进行处理，这个时候最直接的想法就是将这些事件过滤掉，这样做既可以免去对它们进行处理，也可以避免它们对程序其它部分产生影响。因此，处理模式二中我们引入了事件过滤器这个概念。

如果对象安装了事件过滤器，则事件在到达目标对象之前先被事件过滤器截获，进行一些处理之后再交给目标对象，该模式总结为一个图如下：

注意：这里需要区别对待，如果你是使用installEventFilter()函数给目标对象注册事件过滤器，那么该事件过滤器只对该目标对象有效，只有该对象的事件需要先传递给eventFilter()函数进行过滤，然后调用相应的事件处理器进行处理，非目标对象则不受影响。如果你是给程序中唯一的QApplication对象注册事件过滤器，那么该过滤器对程序中的每一个对象都有效，任何对象的事件都是先传给eventFilter()函数，然后再使用事件处理器进行处理。

3、事件处理模式三

打卡汇报 (276)

评论排行

打卡汇报 (2)
学习 (1)
如何学习黑客技术 (1)
解析Qt自带的Style示例 (0)
QPainter (0)
qt事件处理 (0)
Qt自定义窗口事件 (0)
智能指针使用-share_ptr (0)
Qt：QListWidget的item (0)
QT的model-view机制 (0)

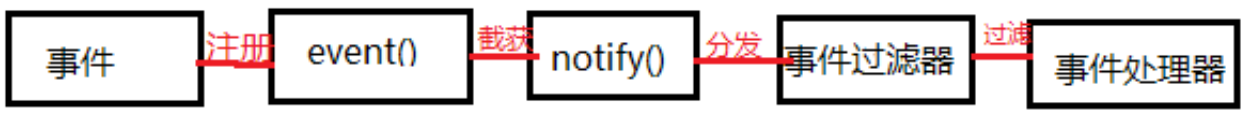
推荐文章

* Nginx 中 HTTP 模块初始化
* 3D语音天气球（源码分享）——完结篇
* 2014年终总结——我的匆匆这一年
* Linux系统中修改用户名的两种方案整理
* C++、Java、JavaScript中的异常处理(Exception)

最新评论

如何学习黑客技术
xiaofan428: 赞一个！请问一下有什么书籍或者软件可以提供一下吗？
学习
lpmygod: http://zhidao.baidu.com/link?url=4Om6dIGSESByHDMNQ...
打卡汇报
lpmygod: 今天晚上陪桥友打桥牌，没有跑步。。，明天卡耐基口才培训，祝好运！！！
打卡汇报
lpmygod: 今天参加了银联数据的面试，感觉很不好，还是一如既往的紧张，看来自己需要作出巨大的改变。正所谓习惯决定...

Qt调用QApplicaton来发送一个事件。所以我们可以重新实现QApplication中的notify()函数来获取事件并进行处理，而且使用该函数获取事件的时间要早于事件过滤器获取事件的时间。但是使用事件过滤器较为简便，因为我们可以有多个事件过滤器，但是只能有一个notify()函数。
用一个图来总结该模式就是：



二、Qt中事件处理的方法

从以上三个处理模式我们可以看出，这是一个不断完善　　个模式的讨论中我们不难找到可以进行事件处理的地方，而这几个地方就是我们在编写程序的时候可以控制事件处理的地方。

1、event()函数

首先是控制事件分发的event()函数，我们可以改写该函数，改变事件的分发方式，这样就可以改变事件处理的结果。

2、notify()函数

实现该函数可以截获事件，并对事件加以处理，但是该方法很少用，这里不做介绍。

3、事件过滤器

实现自己的事件过滤器就可以改变事件处理的方法和结果，这个方法比较常用。

4、事件处理器

事件处理的最后一步，也是最重要的一步就是事件处理器，因为它才是真正进行事件处理的地方，我们可以改写以有的事件处理器，以此改变已有事件的处理方法和处理结果，我们也可以定义自己的事件类型和相应的事件处理器。

注意：以上四种方法中最常用的是后两者：事件过滤器和事件处理器。

补充内容：

1、事件的传递

包括鼠标和键盘事件在内的很多事件都可以被传递。如果事件在到达目标对象之前没有被截获处理，或者已经传递给了它的目标对象但目标对象并没有进行处理，那么此时，目标对象的父对象将变成新的目标对象，整个事件处理的过程将重复进行，直到该事件被处理或者到达最顶层对象为止。

2、**event**实例解析

下面的代码是QWidget::event()函数的代码，从代码中可以看出event()函数确实只进行事件的分发而不负责事件的处理。由于函数代码过多，且都是一类型的用switch语句进行处理的，这里只贴出一部分代码：

```cpp
bool QWidget::event(QEvent *event)
{
    Q_D(QWidget);

    // ignore mouse events when disabled
    if (!isEnabled()) {
        switch(event->type()) {
        case QEvent::TabletPress:
        case QEvent::TabletRelease:
        case QEvent::TabletMove:
        case QEvent::MouseButtonPress:
        case QEvent::MouseButtonRelease:
        case QEvent::MouseButtonDblClick:
        case QEvent::MouseMove:
        case QEvent::ContextMenu:
#ifndef QT_NO_WHEELEVENT
        case QEvent::Wheel:
#endif
            return false;
        default:
            break;
        }
    }
    switch (event->type()) {
```

更多相关资源：　qt

```
    case QEvent::MouseMove:
        mouseMoveEvent((QMouseEvent*)event);
        break;

    case QEvent::MouseButtonPress:
        // Don't reset input context here. Whether reset or not is
        // a responsibility of input method. reset() will be
        // called by mouseHandler() of input method if necessary
        // via mousePressEvent() of text widgets.
#if 0
        resetInputContext();
#endif
        mousePressEvent((QMouseEvent*)event);
        break;

    case QEvent::MouseButtonRelease:
        mouseReleaseEvent((QMouseEvent*)event);
        break;

    case QEvent::MouseButtonDblClick:
        mouseDoubleClickEvent((QMouseEvent*)event);
        break;
#ifndef QT_NO_WHEELEVENT
    case QEvent::Wheel:
        wheelEvent((QWheelEvent*)event);
        break;
#endif
#ifndef QT_NO_TABLETEVENT
    case QEvent::TabletMove:
    case QEvent::TabletPress:
    case QEvent::TabletRelease:
        tabletEvent((QTabletEvent*)event);
        break;
```

在上一篇中我们了解了Qt中事件处理的方式，也提到了最常用的就是使用事件处理器和事件过滤器这两种方法。在这一篇，我们就来看看事件处理器和事件过滤器是怎么使用的。

一、事件处理器使用实例

Qt中针对每一种常见的事件类型都提供了相应的事件处理器，我们如果想捕获某种类型的事件并进行自定义处理，那么只需要实现重写这些事件处理器就行，至于常见的时间类型和对应的事件处理器如下图：

enum QEvent::Type

This enum type defines the valid event types in Qt. The event types and the specialized classes for each type are as follows:

Constant	Value	Description
QEvent::None	0	Not an event.
QEvent::AccessibilityDescription	130	Used to query accessibility description texts (QAccessibleEvent).
QEvent::AccessibilityHelp	119	Used to query accessibility help texts (QAccessibleEvent).
QEvent::AccessibilityPrepare	86	Accessibility information is requested.
QEvent::ActionAdded	114	A new action has been added (QActionEvent). 对应的事件处理器
QEvent::ActionChanged	113	An action has been changed (QActionEvent).
QEvent::ActionRemoved	115	An action has been removed (QActionEvent).
QEvent::ActivationChange	99	A widget's top-level window activation state has changed.
QEvent::ApplicationActivate	121	The application has been made available to the user.
QEvent::ApplicationActivated	ApplicationActivate	This enum has been deprecated. Use ApplicationActivate instead.
QEvent::ApplicationDeactivate	122	The application has been suspended, and is unavailable to the user.
QEvent::ApplicationFontChange	36	The default application font has changed.
QEvent::ApplicationLayoutDirectionChange	37	The default application layout direction has changed.
QEvent::ApplicationPaletteChange	38	The default application palette has changed.
QEvent::ApplicationWindowIconChange	35	The application's icon has changed.
QEvent::ChildAdded	68	An object gets a child (QChildEvent).

事件类型

在我的程序中，我使用到了鼠标滚轮事件，主要实现的就效果就是大家比较熟悉的：用一个控件显示图片，当滚动鼠标滚轮的时候可以调整图片显示的大小。

这里我要做的就是捕获该图片显示控件的鼠标滚轮事件，然后改写该控件的鼠标滚轮滚动事件处理器。

首先我们来看看鼠标滚轮事件以及相应的事件处理器是啥，查看上面的图即可知道：

QEvent::Wheel	31	Mouse wheel rolled (QWheelEvent).

在Qt帮助文档里面我们可以查看到如下语句“The event handler QWidget::wheelEvent() receives wheel events.”。也就是说最后事件处理器会调用wheelEvent()函数来处理该事件，因此我们需要的就是改写wheelEvent()这个函数。

第一步：在头文件中申明该函数：

```
protected:
    void paintEvent(QPaintEvent *event);
    void dragEnterEvent(QDragEnterEvent *event);
    void dropEvent(QDropEvent *event);
    void dragLeaveEvent(QDragLeaveEvent *event);
    void wheelEvent(QWheelEvent *event);
```

第二步实现wheelEvent()函数(这里主要关注的是事件处理的框架，而不是具体的示例，因此不去过多讲解代码细节)：

```cpp
<span style="font-size:16px;">void ImageWidget::wheelEvent(QWheelEvent *event)
{
    int numDegrees = event->delta();
    int num = numDegrees / 120;
    if(num < 0)
    {
        num = 0 - num;
    }

    if (event->orientation() == Qt::Horizontal) {

        //scrollHorizontally(numSteps);
        scale *= 1.25;
        resize(this->scale * this->size());
    } else {
        //scrollVertically(numSteps);
        if(numDegrees > 0)
        {
            //scale *= 0.75;
            resize(num * 0.75 * this->size());
        }
        else if(numDegrees < 0)
        {
            //scale *= 1.25;
            resize(num * 1.25 * this->size());
        }
    }

    event->accept();
}</span>
```

到这里这个功能就实现了，大家看看上面的头文件就知道这里还实现了其它类型事件的处理，其实都是一样的思路，找到欲处理的事件类型，找到对应的事件处理器，重写事件处理器中处理事件的方法即可。

二、事件过滤器使用实例

Qt事件模型中一项非常强大的功能就是一个QObject实例可以监视另一个QObject实例中的事件，实现方法是在目标对象中安装事件过滤器。这里我们接着上面的实例进行，为上面的图片浏览器增加一个功能：在我的程序中设置了一个自动播放的按键，点击按键后，在图片显示部件中就会显示一系列图片，就像动画播放许多幅图片一样，当在上面点击鼠标左键时，就停止动画播放而是停留在当前播放的图片；当我再点击鼠标右键的时候，又恢复为动画播放模式。

首先说一下主要的页面布局：我在一个QMainWindow中放置一个自定义的图片显示部件。对照上面的说明，我们很容易知道，实现该功能的方法就是在目标部件(自定义的图片显示部件)上注册事件过滤器，此时的事件过滤器就是我们所说的监视对象，完成这些步骤之后，当目标部件有事件产生后，首先会传递给监视对象(事件过滤器)进行处理而不是该事件对应的事件处理器。所以说我们可以截获事件进行处理。监视对象截获目标对象的事件后就会调用自己的eventFilter()函数处理这些事件。

总结起来就两个步骤：

第一：对目标对象调用installEventFilter()来注册监视对象(事件过滤器);

第二：重写监视对象的eventFilter()函数处理目标对象的事件。

我们就严格按照这两步走：

首先使用installEventFilter()函数给目标对象注册事件监听器：

```cpp
<span style="font-size:16px;">//给图片显示部件注册事件过滤器
imageWidget->installEventFilter(this);</span>
```

然后就是重写监视对象eventFilter()函数：

```cpp
<span style="font-size:16px;">//图片显示部件时间过滤器处理
bool PMainWindow::eventFilter(QObject *target, QEvent *event)
{
    if(target == imageWidget)
    {
        qDebug("The imageWidget generate the event!");
        if(event->type() == QEvent::MouseButtonPress)
        {
            QMouseEvent *mouseEvent = static_cast<QMouseEvent *>(event);
            if(mouseEvent->buttons() & Qt::LeftButton)
            {
                qDebug("The Left Button Event!");
                killTimer(timeId);
            }
            else if(mouseEvent->buttons() & Qt::RightButton)
            {
                qDebug("The Right Button Event!");

                //clickNum++;
                //again();
                timeId = startTimer(3000);
            }

            return true;
        }
    }

    //其它部件产生的事件则交给基类处理
    return QMainWindow::eventFilter(target, event);
}</span>
```

上一篇 解析Qt自带的Style示例

下一篇 Qt Graphics View FrameWork图形视图框架

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