20170913

1. 今天在使用addStretch，布局的时候，发现addStretch竟然是可以平均分配的，有意思。比如：



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | QVBoxLayout \*buttonLayout = new QVBoxLayout;  buttonLayout->addStretch(1);  buttonLayout->addWidget(Button1);  buttonLayout->addStretch(1);  buttonLayout->addWidget(Button2);  buttonLayout->addStretch(1);  buttonLayout->addWidget(Button3);  buttonLayout->addStretch(6); |

您会发现，buttonLayout的布局将空白没有widget的地方分成了9份，然后按照您所规定的地方分配弹簧，于是布局起来就方便多了，以前没有发现这个，一直还为布局头疼呢，现在好了点。

# Qt之QLabel

最近在用Qt设计一个小程序，  
想让一幅图片自适应窗口大小

找到了很多方法但都会出一些小问题，  
  
刚刚摸索出解决办法了，在些记录。

思想：

1 显示图像是用QLabel  
2 在QWidget或其他窗口部件上显示（也可直接显示在窗口上，后面会说明为什么不）

现在问题就明朗了，主要就是调整QLabel大小随窗口大小自动变化。

使用方法：

1 在要显示的窗口放置一个QWidget，并使用用布局管理器（以便窗口可以规范布局）

2 在QWidget内放置QLabel ，位置为左上角，不使用任何布局管理器

3 在构造函数内添加如下代码：

ui->label->setScaledContents(true);

4 编辑界面重绘函数，加入如下代码：

ui->label->resize(ui->widget->size());  
即可。  
  
  
原理明：  
当我们调整窗口使其变化时，paintEvent 函数自动被调用执行，  
执行 ui->label->resize(ui->widget->size());语句为：设置label大小为widget大小。

而ui->label->setScaledContents(true);为设置QLabel自动缩放，既：显示图像大

小自动调整为Qlabel大小。

这样就实现了自适应窗口显示图片。  
  
最近因为项目的需要，需要在QLabel标签上显示图片。那么问题就来：1.图片如何自适应窗口控件的大小；2.图片如何随着窗口大小的改变而改变呢？这就是两个比较基本也最常见的问题了。

解决问题一：图片如何自适应窗口控件的大小？

相对于第一个问题，最初的想法的就是QLabel用布局进行管理，这样岂不就是固定了QLabel的大小，如何图片岂不也就是固定在QLabel里面，可是后来尝试了这种方法之后就呵呵（实践永远是真理）。虽然QLabel用布局管理了，但是当QLabel加载了图片以后，由于图片的大小问题，QLabel会压缩其他控件的位置，使得整个布局都变得凌乱了。比如下面的代码：

**[cpp]** [view plain](http://blog.csdn.net/wusiyuan163/article/details/51107574) [copy](http://blog.csdn.net/wusiyuan163/article/details/51107574)

[在CODE上查看代码片](https://code.csdn.net/snippets/1640759)

1. QLabel \*image1Label = **new** QLabel();
2. QLabel \*image2Label = **new** QLabel();
3. image1Label->setPixmap(QPixmap::fromImage(img1));
4. imgae2Label->setPixmap(QPixmap::fromImage(img2));

实践证明这种简单粗暴的方法是解决不了图片自适应窗口控件大小的问题的。那么既然是由于图片大小问题导致的原因，那在向QLabel加载图片的时候，将图片的大小缩变为QLabel的大小那么解决问题了。这样想是正确，图片缩放到QLabel的大小，那再加载到QLabel中的时候，就不会出现QLabe挤压其他空间的现象了。直接上代码看看：

**[cpp]** [view plain](http://blog.csdn.net/wusiyuan163/article/details/51107574) [copy](http://blog.csdn.net/wusiyuan163/article/details/51107574)

[在CODE上查看代码片](https://code.csdn.net/snippets/1640759)

1. QPixmap \*img1 = **new** QPixmap("1.jpg");
2. QPixmap \*img2 = **new** QPixmap("2.jpg");
4. img1->scaled(image1Label->size(), Qt::KeepAspectRatio);
5. img2->scaled(imgae2Label->size(), Qt::KeepAspectRatio);
7. image1Label->setPixmap(img1);
8. imgae2Label->setPixmap(img2);

通过编码实现，好像第一个问题解决了呢。恩，差不多就这样，第一个问题解决了，那么就该解决第二个问题了。第二个问题是什么呢，不用翻回去看问题了，我来告诉你（哈哈）：图片如何随着窗口大小改变而改变呢？

解决问题二：如何使得图片的大小随着窗口大小改变而改变呢？

貌似这个问题好像不存在，因为有人会说：你的QLabel不是在布局管理中吗，那么QLabel的大小就可以调整嘛。对的，是这样的，布局中的QLabel是可以随着窗口大小变化而做出相应的变化的。由于我们采用了上面所说的改变图片尺寸的方法来解决问题一的，那么现在就有个新问题了：当QLabel的大小发生变化的时候，我也得重新改变图片的大小，使其适应新的QLabel的大小。是的，思路就是这样的，没啥偏差，那就编码实现看看效果呗。为了能够实现重绘图片的大小，那就得重新实现窗口的resizeEvent虚函数了。这个虚函数是基类QWidget的虚函数，主要是对窗口尺寸变化事件的一个处理。我们需要对这个函数进行重新实现。在相应的头文件中添加下面的代码：

**[cpp]** [view plain](http://blog.csdn.net/wusiyuan163/article/details/51107574) [copy](http://blog.csdn.net/wusiyuan163/article/details/51107574)

[在CODE上查看代码片](https://code.csdn.net/snippets/1640759)

1. // rewrite virtual function
2. **virtual** **void** resizeEvent(QResizeEvent \*event);

然后在相应的cpp文件中实现这个虚函数，代码是：

**[html]** [view plain](http://blog.csdn.net/wusiyuan163/article/details/51107574) [copy](http://blog.csdn.net/wusiyuan163/article/details/51107574)

[在CODE上查看代码片](https://code.csdn.net/snippets/1640759)

1. void CMViewer::resizeEvent(QResizeEvent \*event)
2. {
3. QWidget::resizeEvent(event);
5. QPixmap \*img1 = new QPixmap("1.jpg");
6. QPixmap \*img2 = new QPixmap("2.jpg");
8. if (image1Widget-**>**height()**>**100)
9. {
10. img1-**>**scaled(image1Label-**>**size(), Qt::KeepAspectRatio);
11. image1Label-**>**setPixmap(img1);
12. }
13. if (image2Widget-**>**height()**>**100)
14. {
15. img2-**>**scaled(imgae2Label-**>**size(), Qt::KeepAspectRatio);
16. imgae2Label-**>**setPixmap(img2);
17. }
18. }

这样下来这个问题貌似得到了解决，可是总觉得怪怪的，不停的要加载图片，改变图片的尺寸大小，虽然在效率上不能看出什么差别，但是不是最好的，所以继续疯狂的在Google上搜索了，后来找到了一个比较好的方法（[这是原文链接](http://blog.csdn.net/yh_1988/article/details/6752733)），下面也分享给大家。

--------------------------------------------------------华丽的分割线--------------------------------------------------------------

思想：

1.部署一个QWidget控件，并用布局管理器进行设计；

2.在QWidget控件上放置一个QLabel，不要使用任何布局。

具体操作过程：

1.在要显示的窗口放置一个QWidget，并使用用布局管理器（以便窗口可以规范布局）；

2 在QWidget内放置QLabel ，位置为左上角，不使用任何布局管理器；

3 在构造函数内添加如下代码：

**[cpp]** [view plain](http://blog.csdn.net/wusiyuan163/article/details/51107574) [copy](http://blog.csdn.net/wusiyuan163/article/details/51107574)

[在CODE上查看代码片](https://code.csdn.net/snippets/1640759)

1. ui->label->setScaledContents(**true**)

4 编辑界面重绘函数，加入如下代码：

**[cpp]** [view plain](http://blog.csdn.net/wusiyuan163/article/details/51107574) [copy](http://blog.csdn.net/wusiyuan163/article/details/51107574)

[在CODE上查看代码片](https://code.csdn.net/snippets/1640759)

1. ui->label->resize(ui->widget->size());

原理解释：

当我们调整窗口使其变化时，paintEvent 函数自动被调用执行，执行 ui->label->resize(ui->widget->size());语句为：设置label大小为widget大小。

而ui->label->setScaledContents(true);为设置QLabel自动缩放，既：显示图像大小自动调整为Qlabel大小。

解释问题：  
1 为什么不直接在窗口上显示QLabel而多加一个QWidget？  
假如我们直接在窗口上显示QLabel，那么有两种情况：  
一、使直接对QLabel使用布局管理器，二、不使用布局管理器。  
  
不使用布局的情况很明显会使窗口布局错乱适应能力着，或者很难获得窗口真实的大小（我用QDockWidget的大小设定时，当窗口锚接入主窗口时种是遮盖图像的一部分）。  
  
第一种情况：直接对QLabel使用布局管理器：那么情况是，打开界面QLabel自动调节为窗口大小，通过拖动使窗口变大后，窗口内有多余的空间后，布局管理器将QLabel自动放大到窗口大小;看似实现了自动适应窗口大小，但当我们想使窗口变小时问题就出来了，窗口无法缩小，原因是布局管理器内的QLabel大小是整个窗口，窗口没已经是最小了。  
  
当使用QWidget做中间介质后，由于QWidget内没有布局管理器，所以当缩小主窗口时其大小可以改变，而QLabel为从QWidget的（0，0）开始绘制，大小为QWidget大小，所以可以实现与窗口同样大小，显示位置也是布局管理器设置的位置。

# 解释问题： 1 为什么不直接在窗口上显示QLabel而多加一个QWidget？ 假如我们直接在窗口上显示QLabel，那么有两种情况： 一、使直接对QLabel使用布局管理器，二、不使用布局管理器。 不使用布局的情况很明显会使窗口布局错乱适应能力着，或者很难获得 窗口真实的大小（我用QDockWidget的大小设定时，当窗口锚接入主窗 口时种是遮盖图像的一部分）。 第一种情况：直接对QLabel使用布局管理器：那么情况是，打开界面 QLabel自动调节为窗口大小，通过拖动使窗口变大后，窗口内有多余的 空间后，布局管理器将QLabel自动放大到窗口大小; 看似实现了自动适应窗口大小，但当我们想使窗口变小时问题就出来了 ，窗口无法缩小，原因是布局管理器内的QLabel大小是整个窗口，窗口 没已经是最小了。 当使用QWidget做中间介质后，由于QWidget内没有布局管理器，所以当 缩小主窗口时其大小可以改变， 而QLabel为从QWidget的（0，0）开始绘制，大小为QWidget大小，所以 可以实现与窗口同样大小，显示位置也是布局管理器设置的位置。

[一去二三里](https://yq.aliyun.com/users/1312259910216341?spm=5176.100239.blogcont24416.2.nhsdH7) 2016-03-12 18:01:00 浏览629 评论0

**摘要：** 简述 QLabel提供了一个文本或图像的显示，没有提供用户交互功能。 一个QLabel可以包含以下任意内容类型： 内容 设置 纯文本 使用setText()设置一个QString 富文本 使用setText()设置一个富文本的QString 图像 使用setPixmap()设置一个图像 动画 使用setMovie()设置一个动画 数字 使用setNum()设置int或double，并转换为纯文本。

简述

QLabel提供了一个文本或图像的显示，没有提供用户交互功能。

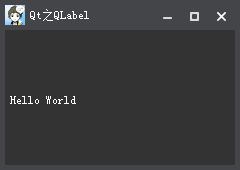
一个QLabel可以包含以下任意内容类型：

| **内容** | **设置** |
| --- | --- |
| 纯文本 | 使用setText()设置一个QString |
| 富文本 | 使用setText()设置一个富文本的QString |
| 图像 | 使用setPixmap()设置一个图像 |
| 动画 | 使用setMovie()设置一个动画 |
| 数字 | 使用setNum()设置int或double，并转换为纯文本。 |
| Nothing | 空的纯文本，默认的，使用clear()设置 |

* [简述](https://yq.aliyun.com/articles/24416#%E7%AE%80%E8%BF%B0)
* [纯文本](https://yq.aliyun.com/articles/24416#%E7%BA%AF%E6%96%87%E6%9C%AC)
  + [显示](https://yq.aliyun.com/articles/24416#%E6%98%BE%E7%A4%BA)
  + [对齐方式](https://yq.aliyun.com/articles/24416#%E5%AF%B9%E9%BD%90%E6%96%B9%E5%BC%8F)
  + [自动换行](https://yq.aliyun.com/articles/24416#%E8%87%AA%E5%8A%A8%E6%8D%A2%E8%A1%8C)
  + [设置行高](https://yq.aliyun.com/articles/24416#%E8%AE%BE%E7%BD%AE%E8%A1%8C%E9%AB%98)
  + [省略](https://yq.aliyun.com/articles/24416#%E7%9C%81%E7%95%A5)
  + [垂直显示](https://yq.aliyun.com/articles/24416#%E5%9E%82%E7%9B%B4%E6%98%BE%E7%A4%BA)
* [富文本](https://yq.aliyun.com/articles/24416#%E5%AF%8C%E6%96%87%E6%9C%AC)
* [图像](https://yq.aliyun.com/articles/24416#%E5%9B%BE%E5%83%8F)
* [动画](https://yq.aliyun.com/articles/24416#%E5%8A%A8%E7%94%BB)
* [数字](https://yq.aliyun.com/articles/24416#%E6%95%B0%E5%AD%97)
* [超链接](https://yq.aliyun.com/articles/24416#%E8%B6%85%E9%93%BE%E6%8E%A5)
  + [方法一](https://yq.aliyun.com/articles/24416#%E6%96%B9%E6%B3%95%E4%B8%80)
  + [方法二](https://yq.aliyun.com/articles/24416#%E6%96%B9%E6%B3%95%E4%BA%8C)
* [总结](https://yq.aliyun.com/articles/24416#%E6%80%BB%E7%BB%93)

纯文本

显示



首先我们构造一个QLabel对象，其中this为其所在的父窗体。通过调用setText可以为标签设置文本（Hello World），这时标签就可以正常显示出来了。为了显示更佳的效果，我们可以通过调用setStyleSheet来设置样式。color: white-顾名思义，就是为标签设置一个文本色（白色）。

QLabel \*pLabel = new QLabel(this);

pLabel->setText("Hello World");

pLabel->setStyleSheet("color: white");

对齐方式

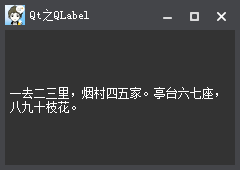
默认的标签文本对齐方式为：左对齐、垂直居中，我们可以通过setAlignment来设置，包括：左、上、右、下、居中对齐，一般情况，我们会进行两两组合（水平方向、垂直方向）。比如：居中对齐

setAlignment(Qt::AlignCenter);

自动换行

如果文本过长，我们可以采用自动换行的方式来显示。

setWordWrap(true);



注意：当使用英文的时候，如果写为类似形式”abcdefghijklmnopqrstuvwxyz”则是不能换行的，why？因为中间没有空格，所以需要写为”abcde fghij klmno pqrst uvwxyz”。

设置行高

一般情况下，自动换行之后文本上下行会距离比较近，我们可以通过下面方式来设置行高。

pLabel->setWordWrap(true);

QString strText = QStringLiteral("一去二三里，烟村四五家。亭台六七座，八九十枝花。");

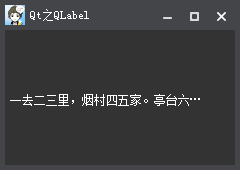
QString strHeightText = "<p style=\"line-height:%1%\">%2<p>";

strText = strHeightText.arg(150).arg(strText);

pLabel->setText(strText);

省略

如果过长，我们又不想换行，只想把其中一部分省略为…，那么我们可以通过QFontMetrics来实现，这里先不介绍QFontMetrics，感兴趣的童鞋可以先自行研究。



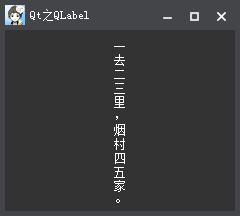
QString strText = QStringLiteral("一去二三里，烟村四五家。亭台六七座，八九十枝花。");

QString strElidedText = pLabel->fontMetrics().elidedText(strText, Qt::ElideRight, 200, Qt::TextShowMnemonic);

pLabel->setText(strElidedText);

垂直显示

默认情况下，文本显示方式为水平方向，如果我们需要在垂直方向上显示，需要用一些小技巧来处理。



QString strText = QStringLiteral("一去二三里，烟村四五家。");

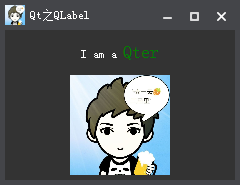
pLabel->setText(strText.split("", QString::SkipEmptyParts).join("\n"));

pLabel->setAlignment(Qt::AlignCenter);

富文本

我们可以在助手中查找关于Using HTML Markup in Text Widgets的资料，查看Qt支持哪些HTML标记。

下面我们来写一段HTML代码，O(∩\_∩)O哈哈~。。。显示不同颜色的文本以及图片



QString strHTML = QString("<html> \

<head> \

<style> \

font{color:white;} #f{font-size:18px; color: green;} \

</style> \

</head> \

<body>\

<font>%1</font><font id=\"f\">%2</font> \

<br/><br/> \

<img src=\":/Images/logo\" width=\"100\" height=\"100\"> \

</body> \

</html>").arg("I am a ").arg("Qter");

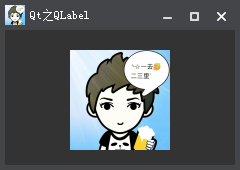
pLabel->setText(strHTML);

pLabel->setAlignment(Qt::AlignCenter);

是不是很神奇？经常我们要用好几个控件来组合才能实现的功能，就仅仅几行HTML代码就搞定了。。。何乐而不为！

图像

首先我们构建一个QPixmap来作为显示的图片，然后设置标签的大小，可以通过setScaledContents按比例缩放图片达到理想的效果。



QPixmap pixmap(":/Images/logo");

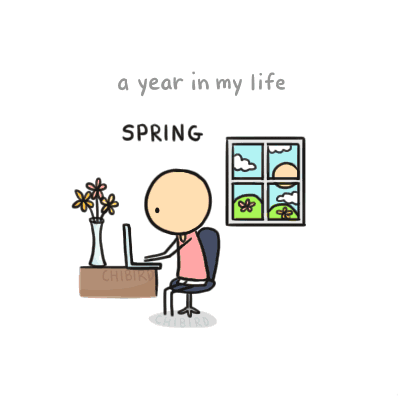
pLabel->setPixmap(pixmap);

pLabel->setFixedSize(100, 100);

pLabel->setScaledContents(true);

动画

这里我们需要使用另外一个类QMovie来控制动画，start()可以进行播放与stop()则可以停止，也可以通过调用setSpeed()来设置动画的播放速度。



QMovie \*pMovie = new QMovie(":/Images/movie");

pLabel->setMovie(pMovie);

pLabel->setFixedSize(135, 200);

pLabel->setScaledContents(true);

pMovie->start();

数字

如果我们需要显示一个数字，则可以调用setNum()，他可以将内容转换为纯文本。

pLabel->setNum(66.6);

很简单就一段代码，我们可以打开源码瞅瞅，究竟setNum是如何实现的。

void QLabel::setNum(int num)

{

QString str;

str.setNum(num);

setText(str);

}

就三行代码，呵呵哒。。。超简单吧，我们要有一颗随时看源码的心！

超链接

我们需要简单使用标签<a></a>写一段简单的HTML超链接代码

方法一

比较简单，直接调用setOpenExternalLinks(true)即可。

pLabel->setText(QString("<a href = \"%1\">%2</a>").arg("http://blog.csdn.net/liang19890820").arg(QStringLiteral("一去丶二三里")));

pLabel->setOpenExternalLinks(true);

方法二

声明一个槽openUrl，将其与linkActivated信号关联。

pLabel->setText(QString("<a href = \"%1\">%2</a>").arg("http://blog.csdn.net/liang19890820").arg(QStringLiteral("一去丶二三里")));

connect(pLabel, SIGNAL(linkActivated(QString)), this, SLOT(openUrl(QString)));

void MainWindow::openUrl(const QString &link)

{

QDesktopServices::openUrl(QUrl(link));

}

总结

通过上面的学习，我们基本将标签的大部分用法都分享了，可以看出HTML、CSS的重要性，所以无论你现在所从事的工作是什么，或者以后做什么，我都建议好好研究下Web，技多不压身。。。上面的内容虽多，但都很简单，也很重要，所以都需要掌握，我们不只是Qter，我们更是工匠-精益求精、严谨、耐心，专注，坚持。