# Qt5-UKUI 开发手册

## 第一章——初识主题框架

### 一、安装

#### PPA安装（未必是最新的）

sudo add-apt-repository ppa:ubuntukylin-members/ukui3.0

sudo apt-get update

sudo apt install qt5-ukui-platformtheme

#### 本地debuild（推荐）

sudo apt install devscripts debhelper

Clone qt5-ukui项目，根据debian/control 下的Build-Depends字段安装编译依赖

执行debuild，安装生成的若干deb包

#### Make安装（不完全安装）

安装编译依赖后，在build目录执行sudo make install

#### 注意事项

要使主题框架生效，我们需要保证应用的QT\_QPA\_PLATFORMTHEHE环境变量为ukui，然而在目前的ukui下默认的环境变量为gtk2，我们需要修改/etc/X11/Xsession.d/99ukui-environment中的对应环境变量为ukui，这个文件由ukui-session-manager提供。

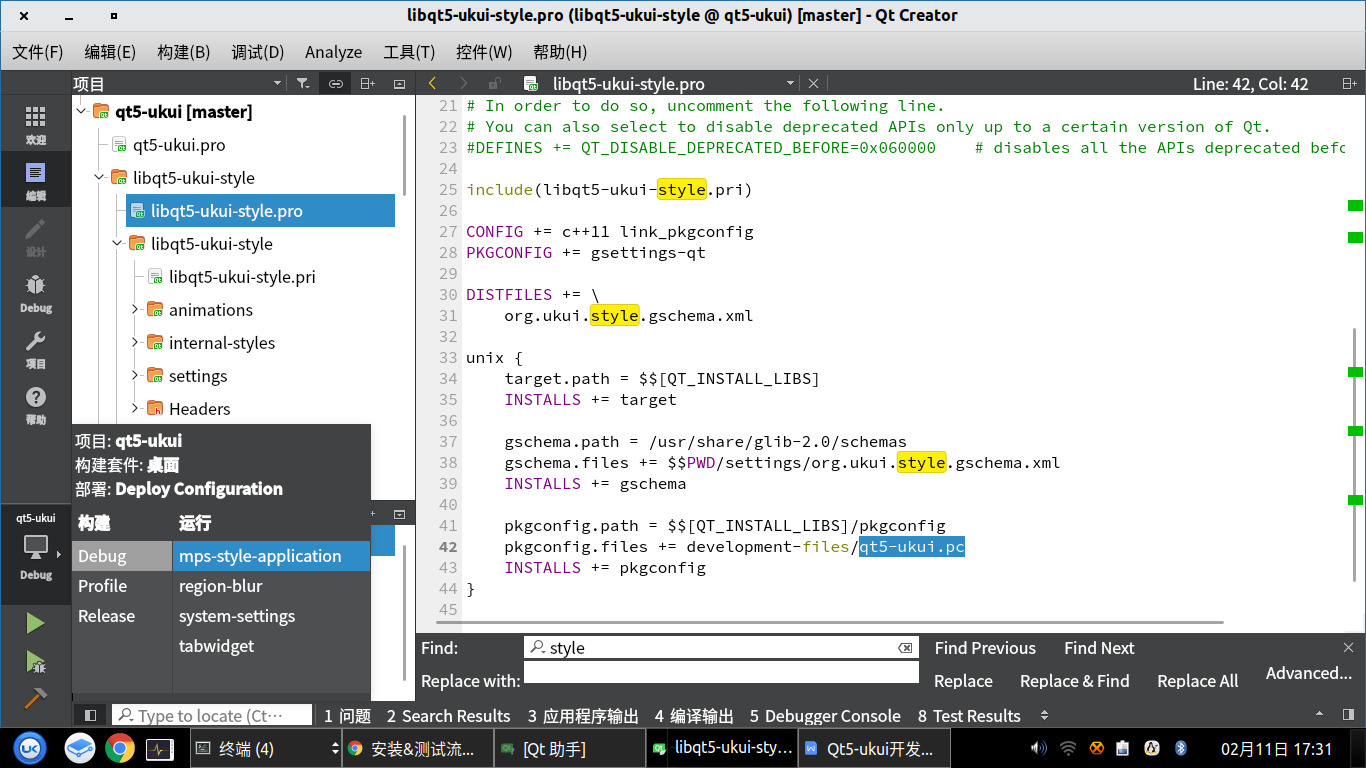
Make安装不完全，一般仅用于快速更新一些so文件，对于gsettings和的dev包的改变则不会同步，如果不能明确自己修改带来的影响，建议使用本地debuild的方式安装

### 测试

在主题框架安装之后，我们可以通过测试用例来对其功能进行测试，判断是否安装成功。

#### 测试用例

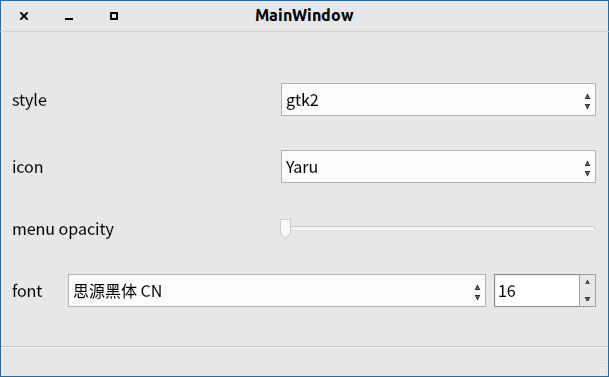
我们使用qtcreator打开项目并且构建之后，能得到关于主题框架各个功能的测试用例。



目前有4个用例，我简述一下它们测试的功能。

#### Systemsettings

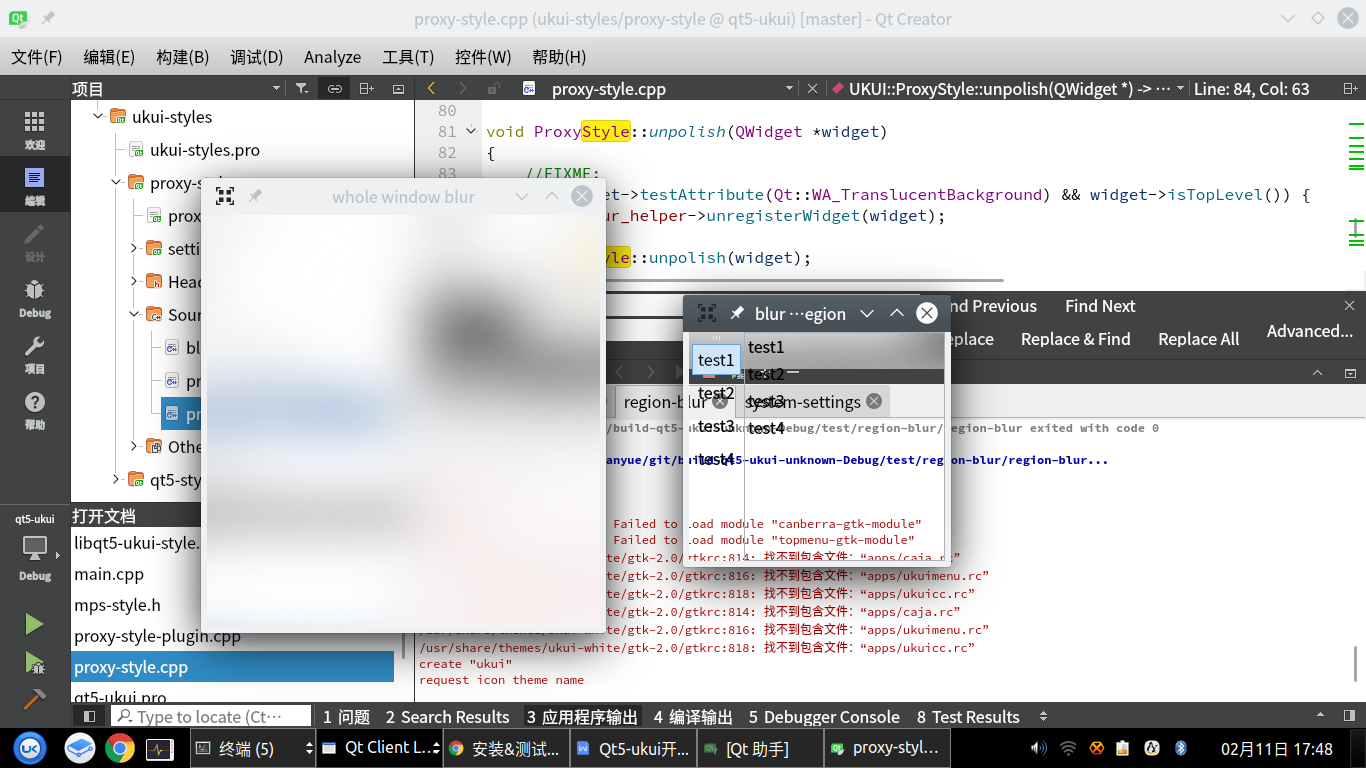
这个用例用来测试主题框架的相关设置及其响应，我通过gsettings使得qt应用能够在主题框架下动态的响应各种设置。



注意Menu-opacity只在ukui主题下生效，我们不方便也没必要对其它主题进行菜单透明度等细节的设置。

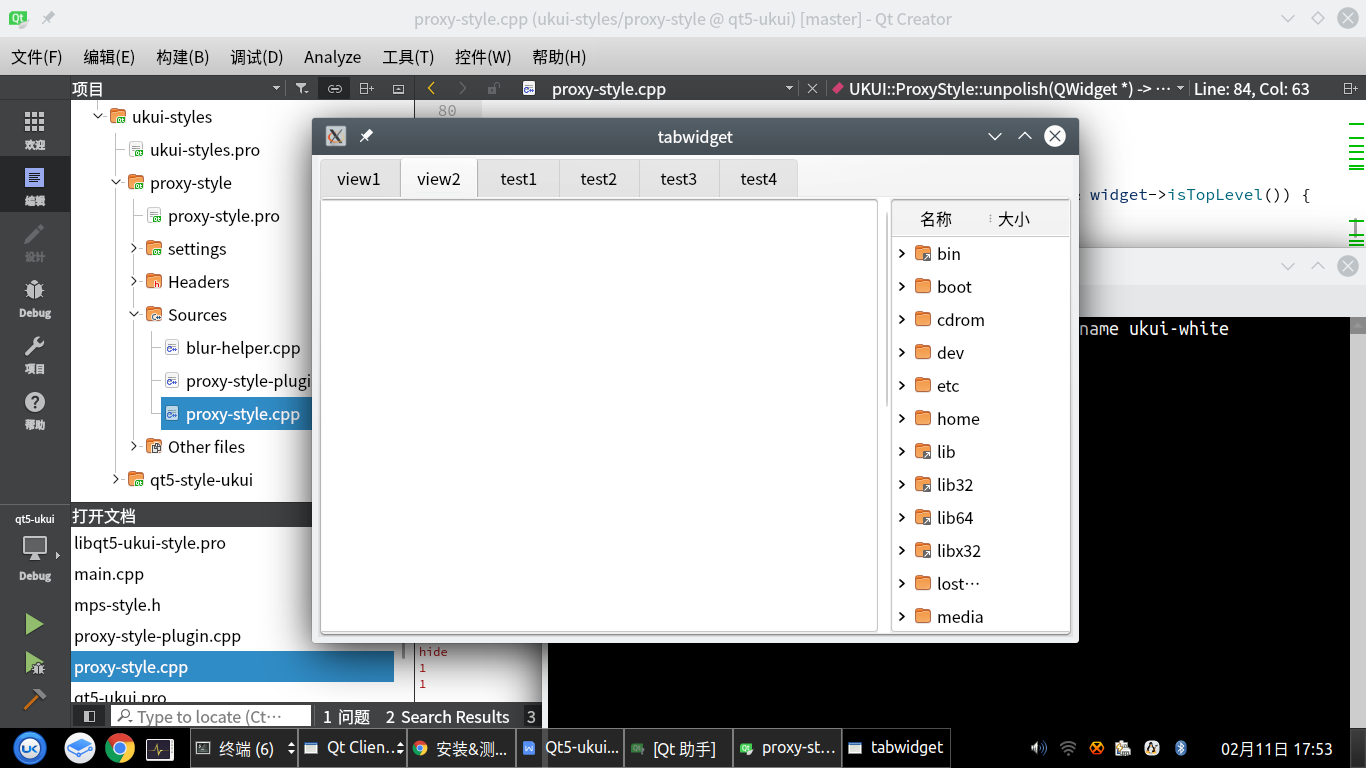
#### Region-blur

这个例子展现毛玻璃效果，但是需要相关服务的支持（由kwin提供），否则只会有透明的效果。



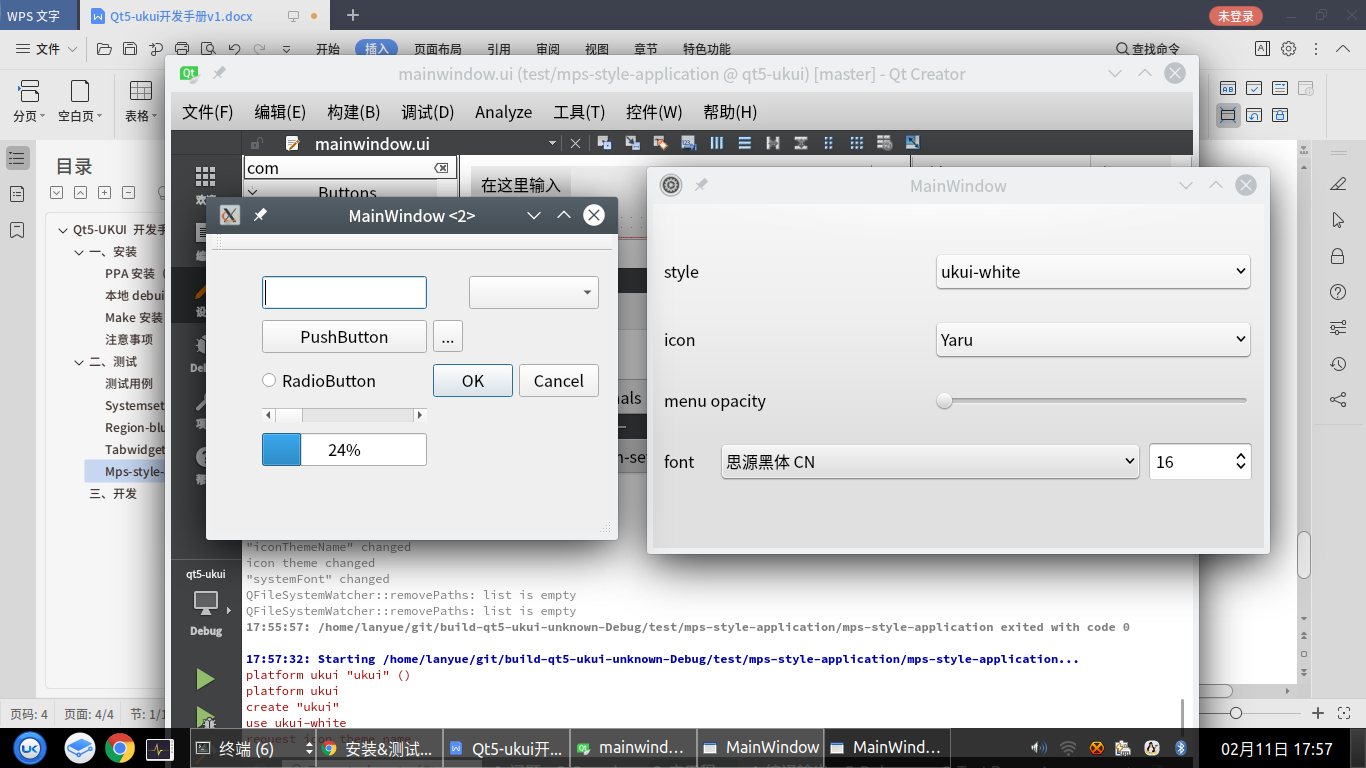
#### Tabwidget

这个例子用来展示ukui主题下标签页切换的动画效果，以此体现基于主题和动画的结合开发与基于控件开发模式的不同。



#### Mps-style-application

这个例子用来展示如何在主题框架下不使用系统主题，在这个例子的窗口样式不会向其它例子一样跟随系统主题的变化而变化，具有一定的独立性。



### 开发思路

我假设大家已经对qt的主题框架有一定的了解，并且能够通过非qss主题+控件的方式进行开发，这里我不会阐述qt主题框架的思路，我欢迎大家针对qt主题框架的理解展开交流，以更好的开发新的主题，但这不是本章节的重点，我想要将的是关于主题框架与目前UKUI项目结合的开发思路。接下来我会从以上两个方面阐述主题框架的开发思路，它们的出发点可能不同但是最后有殊途同归的意思。

要结合主题框架下进行UKUI的应用开发，我们首先得确定开发的应用究竟是否属于系统主题。

#### 使用系统主题——代表应用：文件管理器

系统主题是一个桌面环境中大部分应用都使用的主题，所有应用基于相同的主题开发是桌面环境统一性的必要保障，我们的主题框架在大部分情况下所指的就是系统主题，一个好的系统主题往往能够奠定整个桌面环境的下限。文件管理器peony是非常典型的使用系统主题进行开发的应用，不管是在旧设计稿还是在新设计稿中，它都采用了系统主题+标准控件为基础路线的开发方式，这种开发方式带来的好处非常多，也是目前社区主流的开发模式。

要开发一个基于系统主题的应用，我们首先需要能够明确一个控件规范，它不是针对某一个应用，而是为了提供整个桌面环境下应用样式的设计指导，一个符合系统主题的应用必然是被设计成严格按照控件规范推进的项目。基于系统主题开发的UKUI应用需要进行的工作目前主要有两个部分——应用控件的标准化和系统主题的开发。

所谓应用控件的标准化，就是使控件能够脱离某一固定的绘制流程，完全使用QStyle接口进行绘制，这样的控件能够做到主题无关，从而适应不同风格的主题，我们常用的QPushButton就是很好的例子。如果能够做到控件的标准化，我们开发的应用就能够完全适应不同的qt主题，这样的应用不仅在我们自己的桌面环境中有着良好的体验，还能够直接以统一的风格展现在其它的桌面环境中，控件的标准化是对一个应用质量的考量，如何在符合规范的情况下开发一个尽可能接近设计稿的应用也是我们目前开发中所欠缺的技术，我认为我们的UKUI3.0如果要比以前有所突破，光是从gtk迁到qt是远远不够的，使用qt开发确实能让我们前期的开发进度和效率提升很多，但这并不意味着我们的工作难度就降低了，gtk的高门槛限制了我们深究，但是qt的低门槛却敦促我们不得不去深究，实话说，我们现在的技术积累还远远不够。

系统主题的开发是主题框架中的重中之重，与应用开发对应的，主题的开发也具有规范，并且已经具有明确的接口和众多的开发实例，规范的主题能够广泛的被应用到各个桌面环境中，也就是所谓的换皮，然而换皮也分高端和低端，高端的换皮工作甚至难于应用开发本身，更加具有技术含量。

在gtk3.16以后，gtk的开发者已经将主题的规范完全定死，让我们没有办法去做一些更加高端的操作，就像qss一样，它只能完成它限定的功能，对这些功能之外的需求则无能为力，我认为将功能定死的规范不是一个非常合理的规范，我们的框架必须建立在一个可扩展的框架之上，这样我们才能够持续的改进和迭代。

系统主题的开发不是一个人的工作，而且往往不是独立的，从目前来看，我们的时间非常紧迫，尤其是各个UKUI应用组件的节点非常赶，我本身也是peony项目的负责人，不可能在这么紧的时间内把主题框架和peony的工作全部做完，但是我们的做法是可以转变的，根据目前的实际情况，完成全部目标的唯一途径就是结合应用进行主题的开发，以文件管理器为例，我们的控件本身大部分可以抽象成标准控件+主题的形式，我们将其与设计组的控件规范对照起来就可以把这一部分整合到系统主题中去，这样当我们完成文件管理器的开发的时候，所用到的大部分控件的主题也已经完成了。不过显然光是文件管理器的控件应用的主题并不能代表所有的主题，也许会有一部分的遗漏，如果有更多的应用采取文件管理器的开发模式，代码的重用率就会大大增加，同时缺少的控件也能够得到互补，我们就能够更加高效的进行开发。结合设计稿，我认为文件管理器和控制面板可以作为系统主题的范本应用进行开发，设计师在设计主题规范的同时应该尽量结合这两个应用，使得开发的方向更加集中。

目前主要对系统主题进行开发的人包括我、美燕姐和凯文，我会负责控件的动效，以及主题的规范性保证，美燕姐和凯文则会对控件的静态样式进行编写。

#### 使用internal style——代表应用：开始菜单、任务栏、侧边栏

除了文件管理器为代表的系统主题类应用，在新UKUI3.0的设计稿中，我们还可以看到一种与系统主题风格有明显划分的应用，它们是以开始菜单、任务栏、侧边栏等组件为代表的另一类主题的应用，我们把这类与系统主题风格不同的主题称为“internal style”。

Internal style的主要特点大致如下：

1.它不跟随或未必跟随系统主题变化；

2.使用internal style的应用在控件的结构和逻辑上和其它应用类似，只是样式不同；

3.在多个桌面组件中使用internal style不会破坏桌面的统一性，反而会被认为是一种扩展性。

Internal style本质上也是和系统主题相同的QStyle接口实现，不同的地方在于它的应用范围不是系统层级，也就是说我们无法通过列举系统主题的方式列举出它们，也无法通过切换系统主题的方式改变它们，但是internal style本身可以通过规定协议的方式对自身进行调整，也就是说它更加灵活和自由，比如做成分为黑、白和使用系统主题的方式也是完全ok的。

我们的开始菜单，任务栏和侧边栏显然不能直接使用系统主题进行开发，但它们的风格比较一致，使用internal style的方式进行开发是完全ok的，而且开发和设计的基本思路与使用系统主题开发并无二致，也是标准控件+主题的开发和设计思路。

为了使开始菜单、任务栏和侧边栏等组件能够在系统主题中使用internal style，我的主题框架中规定继承自类名为InternalStyle的类或其自身会被标记为internal style，主题框架会认为它具有以上特性，在切换主题时不会对改应用主题进行切换，它的使用方式可以参考测试中的mps-style-application。

除了系统主题的推进，为了internal style也能够同时推进，我在主题框架中创建了一个MPSStyle的类（MPS是Menu Panel Sidebar的缩写，也有主生产计划的意思），我希望开始菜单、任务栏和侧边栏以及用到了相同风格设计的控件能够以结合主题框架的形式进行开发。

想要在开始菜单、任务栏和侧边栏中共用mpsstyle的办法，在理想的情况下是在主题框架下开发MPSStyle，并由libqt5-ukui-style-dev提供MPSStyle的引用（使用pkg-config，qt5-ukui）。但是考虑到主题框架入库等一些风险性，我们可以采取拷贝代码到自己项目的过渡形式入库，当然这只是为了入库不得已而为之的做法，我们在开发时应该尽量使用主题框架，并协作修改MPSStyle，使其完善成一个能够适应相关应用群的主题。

关于MPSStyle的开发一开始可能由我牵头，但是我希望这一块能够在开发过程中移交给刘号、何朴尧和刘善文。MPSStyle和我的工作关系其实只有“主题”两个字是共同的，交集并不多，在开发过程中我们可以相互交流问题，提供解决方案，但是肯定是没有余裕去帮他人写代码的，我设计internal style的目的就是能够将主题框架的工作合理的分出来，避免出现只有我一个开发者的窘境。

#### 最后——积累和逐步完善的过程

不管是开发应用还是开发主题，都是一个积累和逐步完善的过程，这和技术的积累是一样的。我们的设计稿往往接近于一个完成品，想要接近它是需要阶段的，我认为我们当下最有价值的阶段还不是个性化的设计，而是基础的完善，它体现在桌面环境的统一性上，比如统一的按钮圆角、控件调色板的设置等，个性化的设计也必须要一个好的环境来衬托。当然，并不是说特色和个性化就不重要了，而且有一些放出去的狠话可能还是得说到做到的。

我在开发的同时，总是想着尽力给出某些技术难点的解决方案，事实上一路上我也是这么过来的，花了很多时间做了很多的思考和无果的努力，然后才能觉得对得起自己技术上的成长。我认为“可以做但是没做完”要好过“做完了但是没做到”，两者的价值显然是不同的，当然对于我们来说，最好和最重要的是“能做完能做到”。我们现在要做的肯定还没有做到，而且还没有做完，那么我们首先要保证我们做完时可以做到，如果我们在这方面频繁让步，让做到这个目标得不到保证，那么最后做完的时候肯定也是做不到的。我认为我现在做不到的东西还有很多，我们做不到的东西肯定也不少，但是让做不到的东西变成做得到，然后还能做完，这才是工作的意义和价值所在。

我们现在面临的问题实际上就是我上面所说的情况的一种抽象，实际上是结合实际情况在两者之间进行取舍，取它的最优解，大家可以把我的想法当成一种理想情况，结合自身实际进行调整，我也会在期间尽量为大家提供初期的帮助，等大家熟悉了新的开发模式之后相信大家会喜欢上它的，而且我现在提倡的做法是历史验证过的，而且在转向qt阵营之后，我们或早或晚都应该要走上这一条路。现在多吃一点苦头，走慢一点，走稳一点也是为了少留点坑，以后少加点班，趁着我们开始还没有太久改革的成本还比较低，我们现在的开发模式还是越快改变越好。

## 第二章——入门

主题框架目前正在开发和迭代，处于一个不稳定的阶段，因此我的文档可能会有一部分的内容来不及改动，比如上一个章节的开发思路中内容并不够完全，只给出了一部分的阶段性的思路，当然如果你不去学习相关的概念，可能根本就看不懂上面一章到底讲了什么东西，这一章我会结合代码或者开发的实例，让大家对主题框架有一个基本的概念，在理解了这些概念之后，反过来再去看上一章的内容，可能会清晰很多。当你能够理解并且运用这些思路进行开发的时候，说明你已经入门了。你要明白主题框架不是我们的最终目的，我们不是为框架而创建框架，而是真真切切的为自己在UKUI3.0的开发而创建这样一个框架，我们的最终目的是构建一个完整的桌面体系，主题框架的开发和理念是这个体系中最为基本的一环，跳开这一环进行开发的图形项目不能称之为一个完整的桌面应用，只能称之为应用。

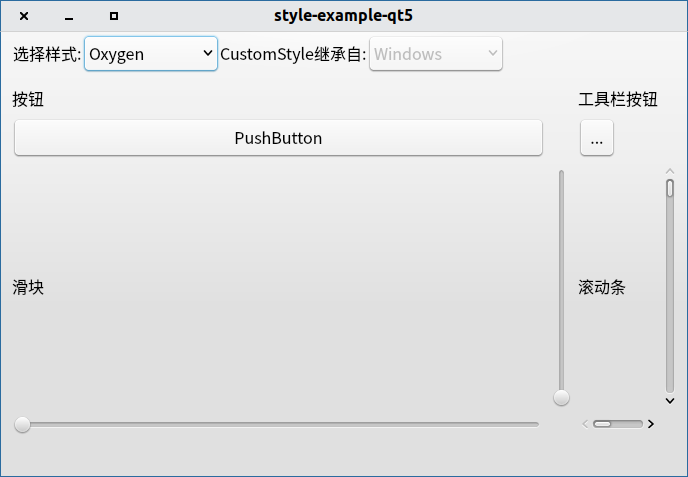
### 示例解析

也许大家都看过我<https://github.com/explorer-cs/style-example-qt5>里面给的例子了，没有也没关系，现在看也不是问题，这些例子每一个都有自己想表达的东西，它们可以看做是主题框架的技术缩影，作为入门例子，它们并不困难，下面我们来一个个的分析它们。

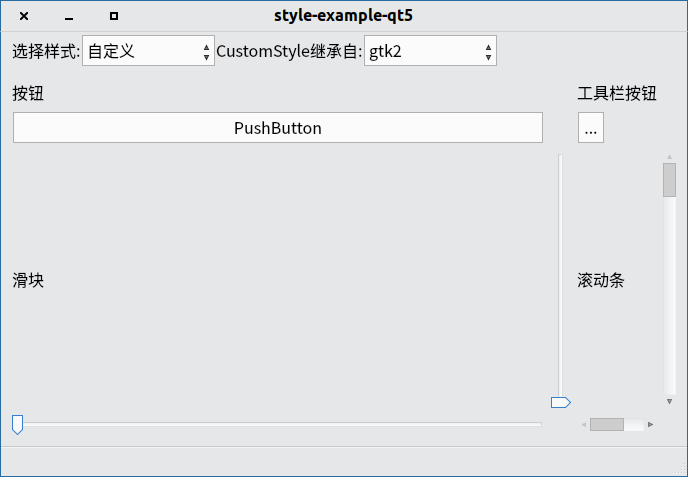
由于篇幅原因，我不会列出代码进行分析，请大家自行切换分支运行示例，并结合代码和注释了解每一个分支想表达的理念。

#### Master分支

所有的例子都是从master分支衍生出来的，master本身是一个可扩展的展示窗口，它允许用户切换窗口的主题，并且提供了一个自定义的主题，可以代理展示其它主题的样式。



选择oxygen作为应用样式



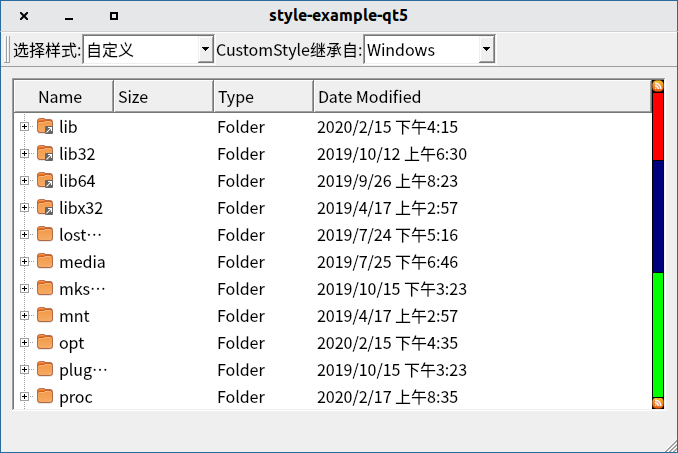
自定义样式的代理样式选择gtk2，展现出来的是gtk2的风格。

主题框架的设计方式就如同这个例子一样，我们首先自定义一个主题，让系统以这个主题作为平台主题，再让这个主题代理显示其它的主题，这样就能够做到qt主题在系统之间的自由切换。

平台主题(platform theme)的概念是在Qt5的Qt5 Platform Abstraction中提出的，它除了能够从整体上管理qt应用的主题之外，还可以进行图标主题，字体等系统配置，这些配置整合到一起，就构成了一个主题框架。

#### 静态样式分支

我们目前的qt主题，都是静态样式的，比如qss，静态样式没有任何的过渡效果，但是大家肯定也对qss这类静态是如何生效的比较感兴趣，我们可以从这个分支的例子中了解整个QStyle的运作机理。



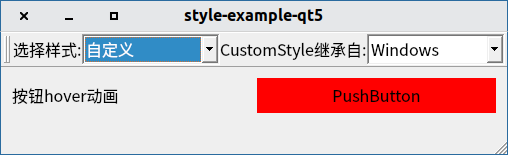
可以看到我们通过对scrollbar相关接口的重写实现了，对scrollbar的样式进行了魔改，在QStyle中，我们上一级传入给我们的参数通常都是有意义的，比如当前子控件的区域，这使得我们能够专注于样式的调整，而不是花精力在计算绘制区域等困难的工作上。

我们的qss本质上也是一个QStyle的实现，只是它具有能够将qss字段解析并且在对应的控件中绘制的能力罢了，QStyleSheetStyle固然是很完善和强大的QStyle，但是它在限定规范的同时也几乎完全抹杀了它的可扩展性，这也是我为什么一直苦口劝说大家抛弃qss的原因，它已经满足不了设计的需求，而且也没有后期扩展的余地了。

言归正传，懂得在合适的情况下运用这些接口进行样式的调整，并且了解接口之间的联动方式是基于主题框架开发模式的开发者所必须掌握的一项技能。我们在开发过程中往往发现，相同的控件，使用系统主题的控件和我们需要的控件在样式上差了很远，我们往往需要修改它的样式，在UKUI的控件规范出炉以后，我们的设计和开发都要往这边靠拢，开发出基于设计规范的主题是我们构建桌面环境的第一步，也是必要的一步，只有在这个规范下，我们的应用才能够得到快速的迭代和开发。规范能够指引设计师在设计的时候不会与开发过于脱节，也能够帮助开发辨别什么才是标准，什么需要特殊处理，这样我们在开发的时候更能有效把控进度，利于我们所重视的逻辑和功能的实现。

#### 动画分支

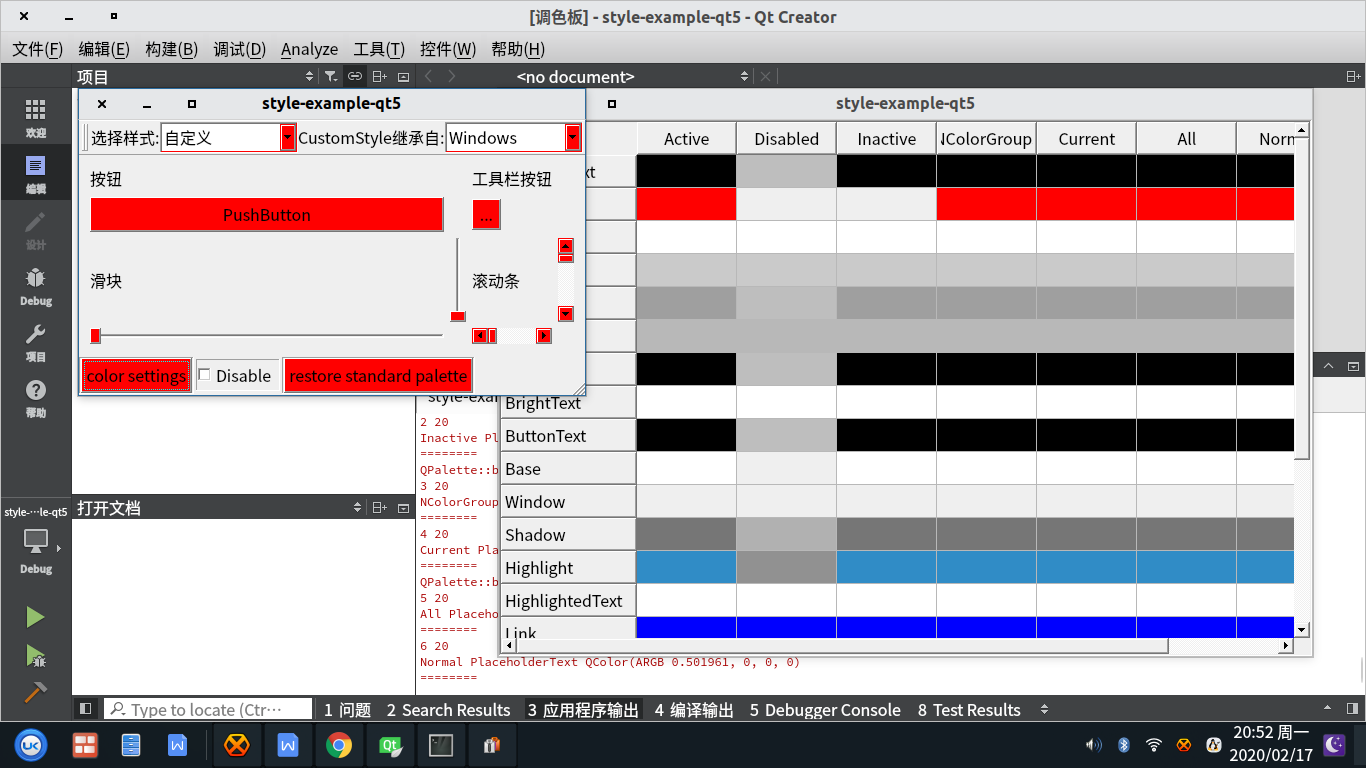
动画是Qt非常重要的一个框架之一，它本身和主题的关联其实并不大，但也能够和主题形成一个非常丰富的交叉学科，如何花最小的代价实现最炫酷的动画是我们今后可能需要花大功夫实现的一点。



这个例子可以说是所有例子中最容易理解的例子了，体现了动画和主题结合的经典套路，但是动画的应用远远不止如此，更多的方式还期待的大家的发掘，相信在有了这些技术的积累之后，大家肯定不会满足于无聊的qss，当然前路还很漫长，让我们一步一步来吧。

#### 调色板分支

从这个例子中大家能够体会QPalette对主题带来的影响。



当然，主题未必遵守QPalette进行绘制。我们早早从上面几个例子看出，怎么绘制一个控件完全可以由我们自己决定，如果愿意的话我们甚至可以自己定一套规范，告诉主题什么控件改用什么颜色绘制，那么我们为什么还需要研究QPalette呢？

试想这样一种情况，我们在控制面板中支持系统主题的切换，也支持控件颜色的切换，当我们使用自己的主题时，可以保证我们的颜色切换机制没有问题，但是如果我们切换到其它系统主题呢？答案是不确定的。使用QPalette作为主题开发的颜色规范是为了使得我们的主题框架能够拥有更广泛的适应性，虽然它可能会带来细微与设计稿在颜色设计上的出入，但它的好处也是显而易见的。

我们在开发的过程中应该很容易发现设计稿的颜色取向和调色板的规范经常会有或多或少的区别，要解决这个问题，从单个主题来看，定义自己的调色板规则我们就能100%做到还原设计稿的颜色样式，但是如果纵观全局，这样带来的后果就是我们失去了调色板在各个主题之间的兼容性，在开发中我们针对这种情况往往需要作出取舍，如果我们希望我们所设计的主题具有更加广泛的应用范围的话，我们应该以遵循规范优先，如果我们脱离这个规范去追求极致的细节，我们则要承担今后转型或者扩展的潜在危险。

从开发者的角度来说，如果设计能够基于qt的规范是最好的，因为这样我们的维护成本能够降低很多，并不是说不能百分之百还原设计稿的颜色，而是从整体考虑出发，我们需要做一个权衡，我认为对于系统主题本身而言，取规范舍细节更加合理。

## 第三章——实战演练

开发离不开项目的需求，如果我只是要大家修改例子效率就太低了，所以实战才是最有意义的，我之所以花时间写这些例子和文档，就是为了大家能够快速投入到实际的开发中来，接下来的任务我们需要当成实战来完成，你的贡献将真正切切的体现在今后系统的每一个应用之上，你需要有极强的警惕性和严谨的态度，虽然你可能只是负责主题框架的一小块，但是只要出现了一丝纰漏，所有的应用可能都会出问题。

### 从例子的衍生开始

从上一张的调色板分支开始衍生一个例子是不错的选择，我们以蓝湖上的控件规范为基准，以控件为单位进行开发，每个人会分配到不同的控件，你的第一步是保证你的控件能够规范化，兼顾设计稿的样式，调色板的规范以及代码的质量。

我们每一个人都可以fork这个项目，当认为自己的控件完成度足够高的时候，我们就可以把它补充到项目的例子中去，这样对后来的新人也是一个非常有帮助的工作。

### 协作开发

当我们的一个控件开发达到要求的时候，我们需要考虑的是将代码集成到主题框架当中来，注意这个时候并不是简单的把代码copy进主题框架的项目里面就可以了，因为主题的协作开发与其它项目的协作开发区别很大，主题的接口具有非常高的重用性，比如绘制按钮的背景的方法也可能会用在绘制其它控件之上，或者说你使用的调色板和主题默认的调色板不一致，这些问题都是及其有可能发生的。你首先要保证你的改动不会影响到全局的展示效果，然后才能够将你的工作整合进来。

这个过程可能一开始是比较艰难的，因为大家都希望自己负责的控件能够百分之百的被还原，但是控件的量一多，必然是会有冲突产生的，这里并不是说是代码上的冲突，它是由于规范的限制或者设计上的漏洞产生的，为了解决这些冲突，我们要相互交流，开发与开发直接，开发与设计之间，设计与设计之间要不断的磨合，达成一个共识，这个共识就是我们主题框架最终的规范。如果我们能够形成这个规范，那么就说明我们已经打通了着一条技术路线，对主题框架已经了如指掌了，这样我们就能够实现技术的飞跃。

当然，这种协同方式也是大家没有接触过的，开发最难的是从0到1的这个过程，这个流程走熟之后很长的一段时间我们都会在快车道上面。

### 主题框架的特性

我总是向大家宣传主题框架的理念，其实主要宣传的还是它的基本理念，这些理念要能够因地制宜、为我所用，还需要结合实际加上一些特性。

#### 特性一——允许独立主题的存在

我在之前第一章和第二章也讲过，允许应用不跟随系统主题，这样带来的好处是自由度的提升，我们在开发过程中可能会遇到与系统主题完全挂不上钩的应用或控件，这个时候强行将它揉合到我们的主题中是不可取的，因为一旦切换其它主题，我们的揉合工作就会失效。

另外一点，即使主题独立开来了，它想要跟随系统主题也并不困难，我们知道主题框架的主题是由gsettings配置的，想要随系统切换只要使用QProxyStyle的特性就可以了，这并不是什么困难的事情。

#### 特性二——根据应用定制调色板

设计这个功能的初衷是和uos的深、浅跟随系统的方式对齐，在主题框架中，我们可以使用定制的调色板，当然这个api还不是特别稳定，还需要进一步的锤炼才能有一个比较成熟的使用方式。

其实我更希望自定义调色板的规则能够由InternalStyle自身决定，我们的系统主题应该尽可能的遵守规范，这样才能保证在切换时风格的兼容，或者在其它qt应用上有良好的表现，如果我们的主题仅仅只是为了自己的应用做定制，那么就跟qss的思路其实没有本质上的区别了，我们同时还得兼顾所有的qt应用。

#### 特性三——事件钩子与特效绑定

其实这不是只有我们的主题才具有的特性，每一个主题都可能会有关于widget的事件钩子或者特效绑定，最典型的就是kde的两大主题breeze和oxygen，对这一块特性的实现我很大程度上参考了这两个主题，事件钩子之后和特效绑定是qstyle的polish机制实现的，有了这个机制，我们能够做到很多意想不到的事情，比如为无边框的窗口提供缩放和拖拽的功能，为半透明窗口提供毛玻璃效果等等。

主题能够做的事可能有限，但是不代表它没有潜力，一些看似不可能的工作，放在主题中可能就变得有可能，这也是主题框架的魅力所在。

### 个人关于主题框架衍生出的一些杂谈

我认为只要具备了相关的知识素养，设计框架对我们而言其实并不困难，但我们应该牢记，我们设计的框架并不是一个悬空的东西，而是切实的为了解决某一个问题而存在的，就好比我们现在的主题框架，它在新UKUI3.0设计稿出来之前，我也绝对不会提出来，因为它那时并不在我的需求内，我用qt5-gtk2-platformtheme效果就很好了，为什么还要自找麻烦呢？

框架的设计都带来的影响是长远的，我们要设计一个框架，首先要能够分析难点，从中梳理出关键的技术点，因为我们要确保有足够的技术理论支撑能够让我们的框架满足我们的需求，如果有一条需求不能够满足，那么这个框架也就没有必要了。

技术的关卡不是一时半会能够攻破的，所以参考现成项目往往是比较合理的，因为它展示出来的东西往往蕴含着技术。在新设计稿出来之后，我很长一段时间没有思路，通过不断的调研社区的项目，终于抓到了头绪，综合了各个社区项目之后，我终于得以动手开发属于我们自己的主题框架了，我认为这个过程虽然漫长，但是却是不可或缺的。在很多时候我们往往会因为项目的压力而放弃一些深层面的思考，因为在没有想清楚之前，这些东西都是没有产出的，我在文件管理器开发较为顺利的情况下，比较幸运的有一段足够长的时间去思考这些问题，当我们做了足够多的准备之后，再动起手来就不会那么的困难了，磨刀不误砍柴工，如果我们的任务总是很紧迫，在有限的时间内没办法完成的很出色，这个时候添加工作量并不能带来太大结果上的转变，我们要想办法让我们的工作从超负荷转化成饱和或者游刃有余的状态，要做到这一点确实很艰难，毕竟现状如此，但是我们不可能一直保持这种状态，主题框架的提出也是一种求变，我希望改变现在各自为政的开发模式，把设计和开发的力量拧成一股绳，这样我们才能够更快更好。

我们今年的发展路线是专注于产品，我们也可以看到越来越多的同事也开始全职或兼职投入到UKUI3.0的项目当中来了，这对于社区的发展无疑是极好的，从长远来看对于公司的将来也会有很大的影响。我认为趁着目前这个人力投入的上升期开始转型还是比较理想的，因为现在不像以前那么缺人，怎么推技术转型都推不动，项目开始的也并不久，不是一个已经成熟到一个不可逆的技术体系，我们缺的可能只是时间，相信在2004展现最好的UKUI3.0是大家的共同愿望，不论用什么样的方法，我们最终的目标都是让UKUI3.0能够获得成功，为了实现这个目标，我希望大家都能够献计献策，哪怕不被采用也没关系，我们对工作的热情和对优麒麟的热爱应该是不变的。