

QSettings读写注册表、配置文件

简述

普通情况下。我们在开发软件过程中，都会缓存一些信息到本地，能够使用轻量级数据库sqlite。也能够操作注册表、读写配置文件。

关于QSettings的使用前面已经介绍过了。比較具体，见“[很多其它参考](#)”，以下介绍下QSettings经常使用功能-读写注册表、配置文件。

- [简述](#)
- [长处](#)
- [读写注册表](#)
 - [一般存储](#)
 - [一般读取](#)
 - [分文件夹存储](#)
 - [替换applicationName](#)
 - [分组](#)
- [读写配置文件](#)
 - [一般存储](#)
 - [一般读取](#)
 - [分组存储](#)
 - [分组读取](#)
 - [分文件夹存储](#)
- [删除内容](#)
- [疑问解释](#)
- [很多其它参考](#)

长处

无需指定注册表路径

普通情况下，我们须要定义一个宏，或者常量字符串来指定保存的注册表位置。

- #define HKEY_CURRENT_USER_QT "HKEY_CURRENT_USER\\SoftWare\\Digia\\Qt"
- const QString HKEY_CURRENT_USER_QT = "HKEY_CURRENT_USER\\SoftWare\\Digia\\Qt";

无需指定配置文件路径

普通情况下，我们须要定义一个宏，或者常量字符串来指定保存的配置文件位置及名称。

- #define INI_QT "C:\\Users\\WangLiang\\AppData\\Roaming\\Digia"
- const QString INI_QT = "C:\\Users\\WangLiang\\AppData\\Roaming\\Digia";

採用以下方式，我们不须要做太多工作。Qt已经非常好的替你实现了！

读写注册表

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

最新随笔

1. MacOS下保护浏览器主页和默认搜索
2. Tokyo Tyrant (TTSer) 系列（一） - 介绍和安装
3. 四：二叉树的镜像递归非递归求解
4. Android 异常 android.os.NetworkOnMainThreadException
5. Android开发学习之路--传感器之初体验
6. Xcode真机调试失败：The identity used to sign the executable is no longer valid
7. hdu 3068 最长回文(manacher & 最长回文子串)
8. bzoj1030【JSOI2007】文本生成器
9. HDU2082母函数模板题
10. Oracle数据库远程连接配置教程

最新评论

1. Re:Android：图解四种启动模式 及 实际应用场景解说 @ 澎湃的代码激情其实作者意思就是默认A activity就是singleTask启动方式了，我读的时候，也默认为这种意思了。只不过作者没有明确的说明，确实存在不严谨的问题。但是，你这出口就骂人也太没...

--nijian81

2. Re:javascript日期相减，求时间差 可以

--韦驮天

3. Re:云环境以下向能耗减少的资源负载均衡方法 您好，冒昧打扰了。请问仿真代码还在吗？最近在学习这个，希望您

一般存储

以下我们以Qt为例。众所周知如今Qt已经属于Digia，也就是说：组织名为Digia，产品名为Qt。

在main()函数中，首先设置组织名、产品名。

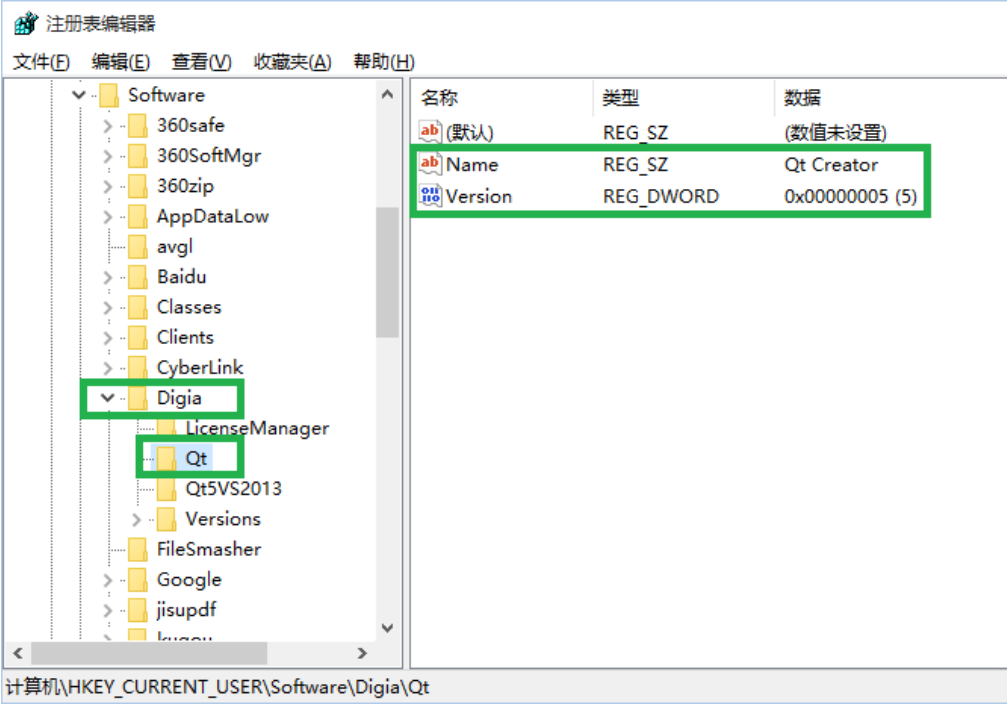
```
QCoreApplication::setOrganizationName(QString("Digia"));
QCoreApplication::setApplicationName(QString("Qt"));
```

然后使用QSettings对注册表进行操作：

```
QSettings settings(QSettings::NativeFormat, QSettings::UserScope,
QCoreApplication::organizationName(), QCoreApplication::applicationName());

settings.setValue("Name", "Qt Creator");
settings.setValue("Version", 5);
```

这时，我们打开注册表regedit。数据就生成了。



一般读取

存储完数据之后，默认的程序启动时须要载入相应的数据。

```
QString strName = settings.value("Name").toString();
int nVersion = settings.value("Version").toInt();
//Name:Qt Creator Version:5
```

这时，我们能够通过查看应用程序输出窗体得到输出结果。

分文件夹存储

假设我们须要在同一路径下建立多个子文件夹该怎么办，以下介绍两种方式。

替换applicationName

如上，我们能够看出。organizationName相应的注册表路径为 HKEY_CURRENT_USER\\SoftWare\\Digia。applicationName相应的为其下一级的文件夹。那么分文件夹就须要更改其相应的applicationName。

能提供一下这个例子的demo，非常感谢。1390231569@qq.com
--gb2312-gbk
4. Re:QSettings读写注册表、配置文件
写的不错，良心
--阿进的写字台
5. Re:Java异步NIO框架Netty实现高性能高并发
兄弟，你的配图一个都没有上传。排个版加上图上传可以？
--今晚打代码

阅读排行榜

1. Python随机数生成方法(48321)
2. Python读取键盘输入(42707)
3. Android：图解四种启动模式及实际应用场景解说(42234)
4. MySQL 性能调优的10个方法(37768)
5. python解压压缩包的几种方法(33188)

评论排行榜

1. Android：图解四种启动模式及实际应用场景解说(2)
2. redis cluster 集群重新启动关闭(2)
3. Java异步NIO框架Netty实现高性能高并发(1)
4. MySQL 性能调优的10个方法(1)
5. 堆排序（C语言实现）(1)

推荐排行榜

1. MySQL 性能调优的10个方法(3)
2. Android：图解四种启动模式及实际应用场景解说(2)
3. python解压压缩包的几种方法(1)
4. QSettings读写注册表、配置文件(1)
5. 最全Pycharm教程（10）——Pycharm调试器总篇(1)

```
QSettings settings(QSettings::NativeFormat, QSettings::UserScope,
QString("%1\\%2").arg(QCoreApplication::organizationName()).arg(QCoreApplication::applicationName()), "Qt5.5");

settings.setValue("Name", "Qt Creator");
settings.setValue("Version", "5.5");

QSettings
settings2(QString("%1\\%2").arg(QCoreApplication::organizationName()).arg(QCoreApplication::applicationName()), "Qt5.6");

settings2.setValue("Name", "Qt Creator");
settings2.setValue("Version", "5.6");
```

分组

替换applicationName的方式看起来有些繁琐，相比之下，使用group分组则会更简单！

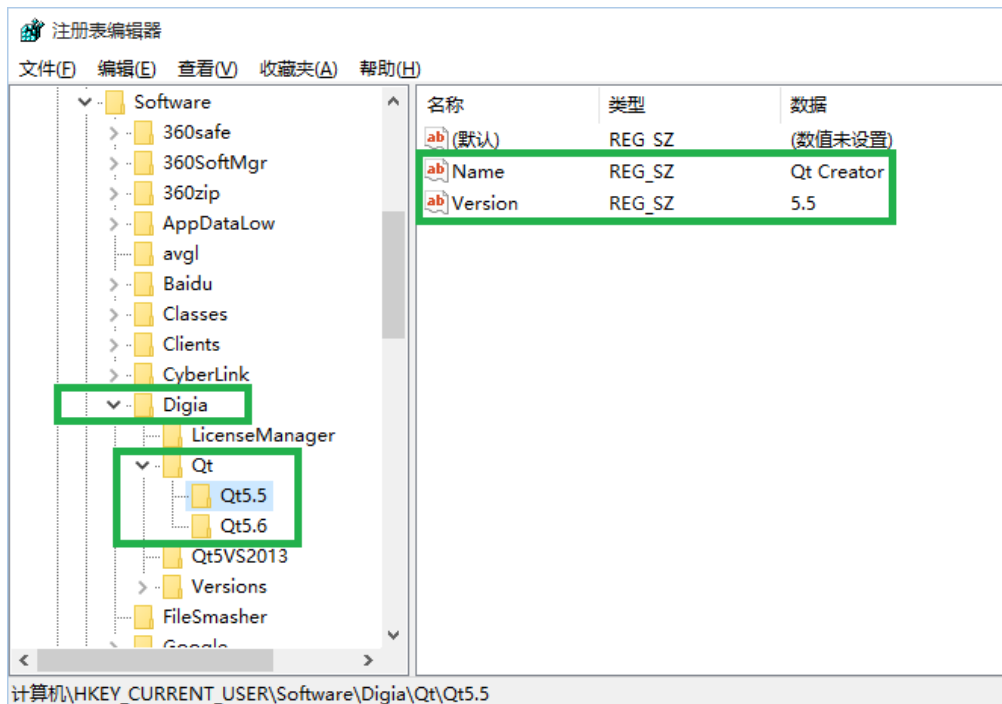
```
QSettings settings;
settings.beginGroup("Qt5.5");
settings.setValue("Name", "Qt Creator");
settings.setValue("Version", "5.5");
settings.endGroup();

settings.beginGroup("Qt5.6");
settings.setValue("Name", "Qt Creator");
settings.setValue("Version", "5.6");
settings.endGroup();
```

这时。我们再次查看注册表数据。

注：

新建文件夹，则须要又一次打开注册表，假设新加入设置，则不须要又一次打开注册表，仅仅须要来回切换相应的选项就可以。



读写配置文件

一般存储

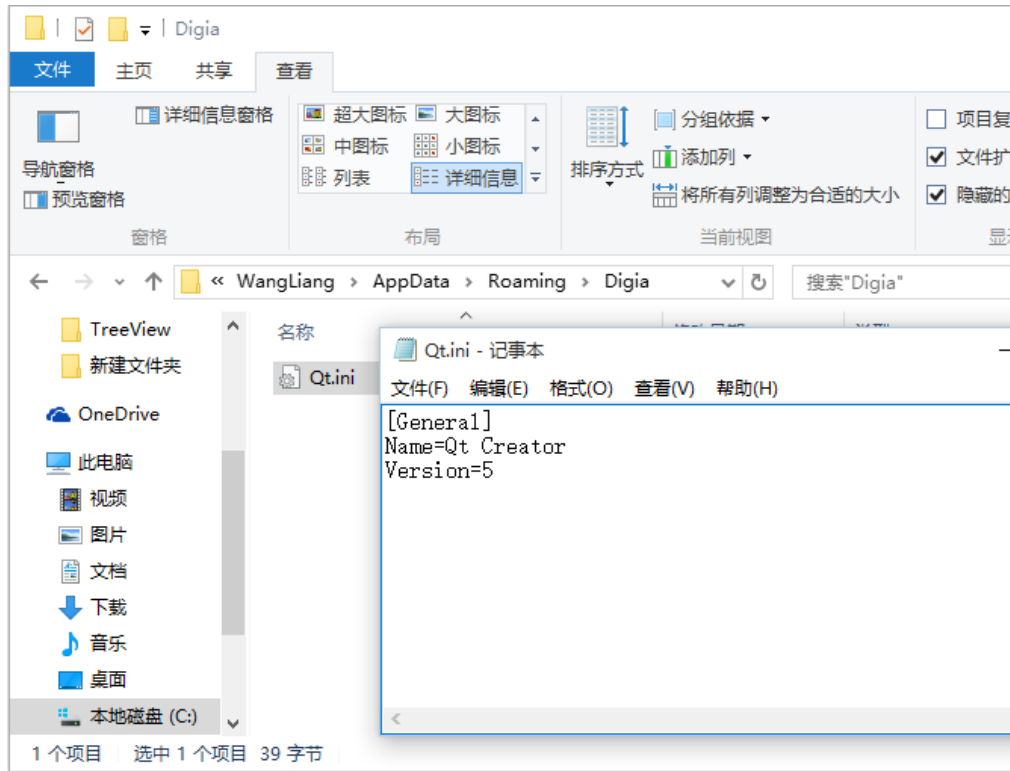
如上，我们仅仅须要将格式从NativeFormat变为IniFormat就可以：

```
QSettings settings(QSettings::IniFormat, QSettings::UserScope,
QCoreApplication::organizationName(), QCoreApplication::applicationName());

settings.setValue("Name", "Qt Creator");
settings.setValue("Version", 5);
```

这时。我们打开相应的存储文件夹。数据就生成了。

我们能够进入文件夹：C:\Users\WangLiang\AppData\Roaming（AppData默觉得隐藏文件，须要设置显示才可查看）。能够看到生成了文件夹“Digia”以及配置文件“Qt.ini”。



一般读取

存储完数据之后，默认的程序启动时须要载入相应的数据。

```
QString strName = settings.value("Name").toString();
int nVersion = settings.value("Version").toInt();
//Name:Qt Creator Version:5
```

这时。我们能够通过查看应用程序输出窗体得到输出结果。

分组存储

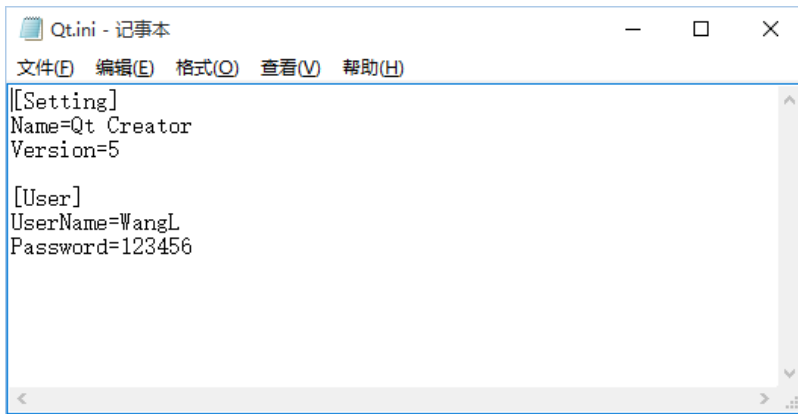
我们能够看到配置文件里包括默认的分组为：General。通常情况下，我们须要对配置进行归类。比如：username、password等信息属于用户组，产品名称、版本属于设置组。

```
QSettings settings(QSettings::IniFormat, QSettings::UserScope,
QCoreApplication::organizationName(), QCoreApplication::applicationName());

settings.beginGroup("Setting");
settings.setValue("Name", "Qt Creator");
settings.setValue("Version", 5);
settings.endGroup();

settings.beginGroup("User");
settings.setValue("UserName", "WangL");
settings.setValue("Password", "123456");
settings.endGroup();
```

这时我们再次查看配置文件，里面已经生成了另外两个分组。



分组读取

```
settings.beginGroup("Setting");
QString strName = settings.value("Name").toString();
int nVersion = settings.value("Version").toInt();
settings.endGroup();
//Name:Qt Creator Version:5

settings.beginGroup("User");
QString strUserName = settings.value("UserName").toString();
QString strPassword = settings.value("Password").toString();
settings.endGroup();
//UserName:WangL Password:123456
```

分文件夹存储

什么时候须要分文件夹存储呢？QQ大家都用过吧，是不是每个用户都有一个相应QQ号的文件夹呢，里面保存各个用户相应的信息。

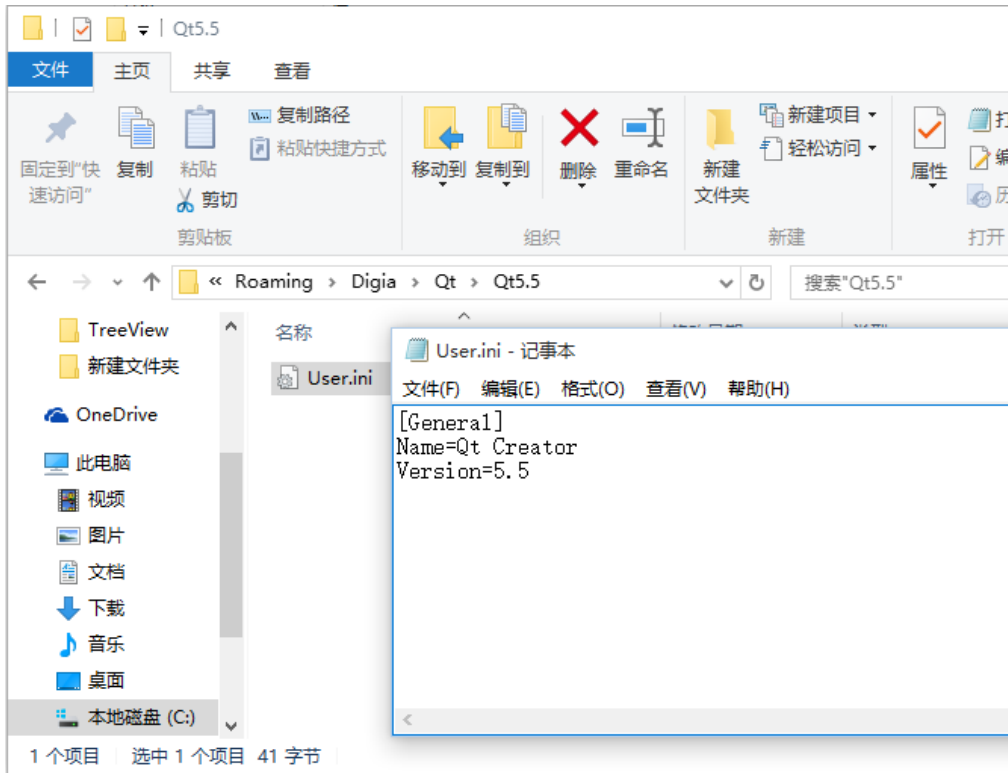
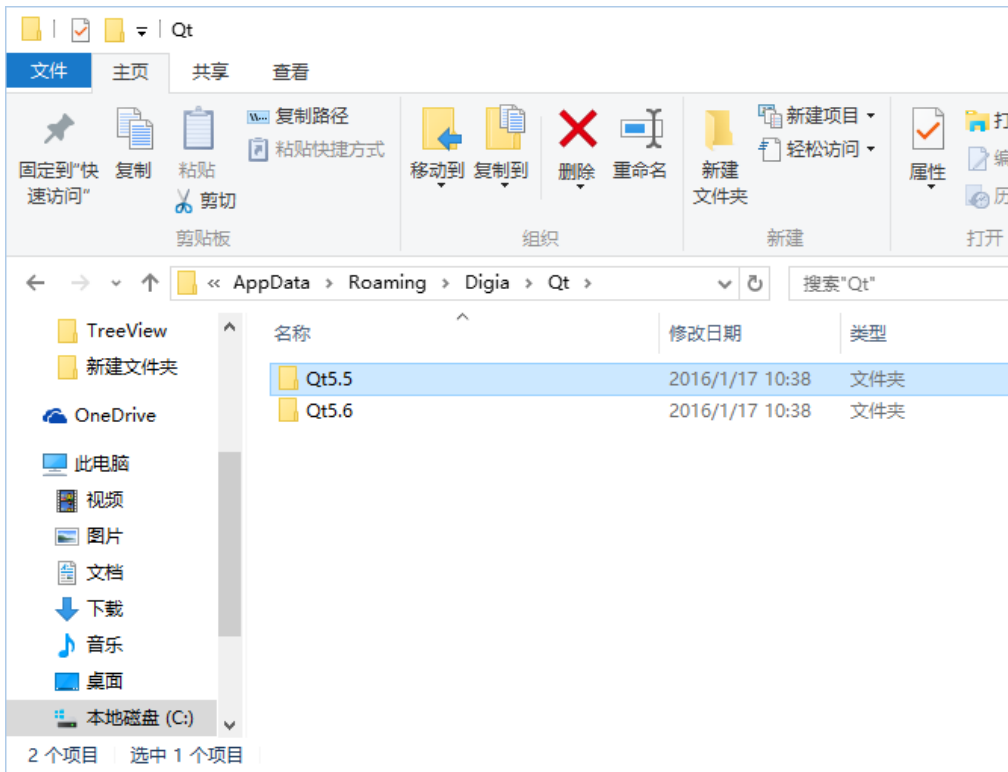
如上，我们能够看出，organizationName相应的本地路径为

C:\Users\WangLiang\AppData\Roaming\Digia，applicationName相应的为其下的配置文件，那么假设我们须要在同一路径下建立多个文件夹就须要更改相应的organizationName，配置文件名则须要更改其相应的applicationName了。

```
QSettings settings(QSettings::IniFormat, QSettings::UserScope,
QString("%1\\%2\\%3").arg(QCoreApplication::organizationName()).arg(QCoreApplication::
applicationName()).arg("Qt5.5"), "User");
settings.setValue("Name", "Qt Creator");
settings.setValue("Version", "5.5");

QSettings settings2(QSettings::IniFormat, QSettings::UserScope,
QString("%1\\%2\\%3").arg(QCoreApplication::organizationName()).arg(QCoreApplication::
applicationName()).arg("Qt5.5"), "User");
settings2.setValue("Name", "Qt Creator");
settings2.setValue("Version", "5.6");
```

这时。我们再次查看本地文件，则会发现C:\Users\WangLiang\AppData\Roaming\Digia\Qt所在文件夹下会生成两个文件夹“Qt5.5”和“Qt5.6”，而且每个文件夹底下会生成相应的配置文件User.ini。



删除内容

删除一个指定的键

```
QSettings settings;  
settings.setValue("Name", "Qt Creator");  
settings.setValue("Version", 5);  
  
settings.remove("Name");  
  
QStringList keys = settings.allKeys();  
// keys: ["Version"]
```

清空全部键

```
settings.clear();
QStringList keys = settings.allKeys();
// keys: []
```

删除设置键以及子设置键

```
QSettings settings;
settings.setValue("Qt5.6", "5.6");

settings.beginGroup("Qt5.5");
settings.setValue("Name", "Qt Creator");
settings.setValue("Version", "5.5");
settings.endGroup();

settings.beginGroup("Qt5.6");
settings.setValue("Name", "Qt Creator");
settings.setValue("Version", "5.6");
settings.endGroup();

settings.remove("Qt5.6");

QStringList strList = settings.allKeys();
// keys: ["Qt5.5/Name", "Qt5.5/Version"]
```

假设key为空字符串，在当前group()的全部键将被删除。

```
QSettings settings;
settings.setValue("Qt5.6", "5.6");

settings.beginGroup("Qt5.5");
settings.setValue("Name", "Qt Creator");
settings.setValue("Version", "5.5");
settings.endGroup();

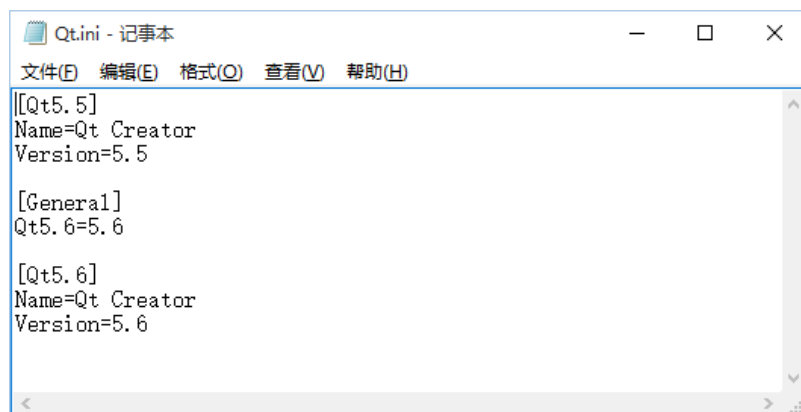
settings.beginGroup("Qt5.6");
settings.setValue("Name", "Qt Creator");
settings.setValue("Version", "5.6");
settings.endGroup();

settings.beginGroup("Qt5.6");
settings.remove("");
settings.endGroup();

QStringList keys = settings.allKeys();
// keys: ["Qt5.5/Name", "Qt5.5/Version"]
```

疑问解释

如上文代码，我们能够知道未删除之前keys: ["Qt5.6", "Qt5.5/Name", "Qt5.5/Version", "Qt5.6/Name", "Qt5.6/Version"], 当中Qt5.6所在分组为默认的General。



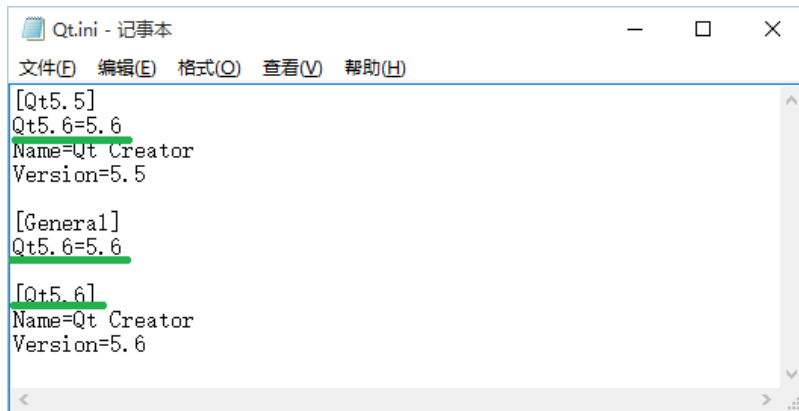
那么调用以下代码：

```
settings.beginGroup("Qt5.6");  
settings.remove("");  
settings.endGroup();
```

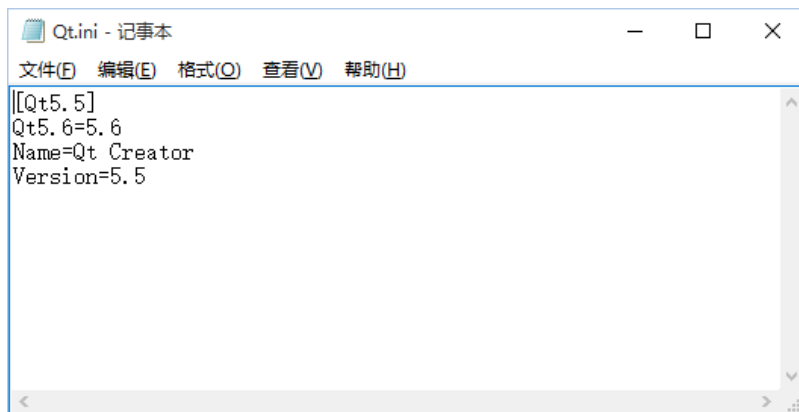
应该仅仅删除相应组中的全部键才对。也就是说剩余的keys应该为 ["Qt5.6", "Qt5.5/Name", "Qt5.5/Version"], General下的键为什么会被删除呢? 好, 这里先跳过, 继续。

General下的键既然能够被删除, 那么在Qt5.5分组下建立相应的Qt5.6键值。应该也会被删除。

删除前:



删除后:



什么鬼。为嘛Qt5.5分组下的Qt5.6相应的键还在呢?

我们继续分析:

删除前: keys: ["Qt5.6", "Qt5.5/Qt5.6", "Qt5.5/Name", "Qt5.5/Version", "Qt5.6/Name", "Qt5.6/Version"]。

助手中关于remove()的说明为: Removes the setting key and any sub-settings of key.

也就是说: Qt5.5/Qt5.6键中即使存在Qt5.6, 可是所属的setting key为Qt5.5而非Qt5.6, 所以不会被删掉。

既然如此, 那么我们的疑问也就不复存在了。

很多其它参考

- [QSettings介绍](#)

【众安尊享e生】 - 国民百万医疗保险，投保详解及案例分析, 每年最低112元——马云杀手锏

好文要顶

关注我

收藏该文

[claireyuancy](#)

关注 - 0

粉丝 - 10

±加关注

1

0

« 上一篇: [后端分布式系列：分布式存储 - HDFS 架构解析](#)

» 下一篇: [【JAVA】两点经纬度直线距离的计算](#)

posted @ 2017-06-29 16:39

claireyuancy

阅读(3879)

评论(1)

编辑

收藏

评论列表

#1楼

2019-03-01 15:17

阿进的写字台

写的不错，良心

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](#)。

- 【推荐】超50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】腾讯云海外1核2G云服务器低至2折，半价续费券限量免费领取！
- 【推荐】阿里云双11返场来袭，热门产品低至一折等你来抢！
- 【推荐】天翼云双十一翼降到底，云主机11.11元起，抽奖送大礼
- 【推荐】流程自动化专家UiBot，体系化教程成就高薪RPA工程师
- 【活动】京东云服务器_云主机低于1折，低价高性能产品备战双11
- 【优惠】七牛云采购嘉年华，云存储、CDN等云产品低至1折

至强®可扩展平台
云计算核“芯”引擎

云主机低至0.9折

年终采购季 错过等明年

立即购买

- 相关博文:
- [MFC INI文件读写](#)

· [C#中配置文件的使用](#)

· [Java 读写Properties配置文件](#)

· [App.Config详解及读写操作](#)

· [.Net MVC 网站中配置文件的读写](#)

» 更多推荐...
- 精品问答: [Java 技术 1000 问](#)
- 云服务器全球购

立即抢购>>

海外1核2G服务器低至2折 半价续费券限量免费领取
- 最新 IT 新闻:

· [全球通吃5G！高通发布最新骁龙865、7系芯片，小米、OPPO将首发](#)

· [报告：美国六大科技巨头十年来在全球避税1000亿美元](#)

· [树莓派4固件更新：大幅降低功耗并提升系统运行速度](#)
- https://www.cnblogs.com/claireyuancy/p/7095249.html
- 9/10

- 仿生芯片可再现生物神经元行为
- 默认情况下, 80% 的 Android 应用正在使用加密流量
- » 更多新闻...



Copyright © 2019 claireyuancy
Powered by .NET Core 3.0.1 on Linux