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。

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

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。

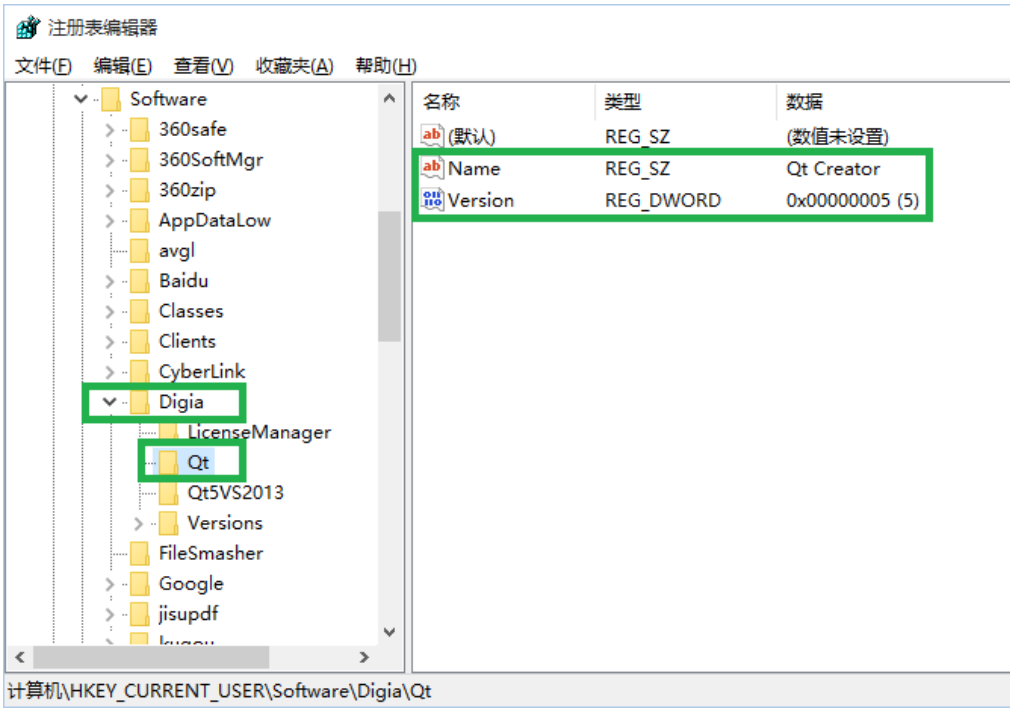
在main()函数中，首先设置组织名、产品名。

```
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

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

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

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

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

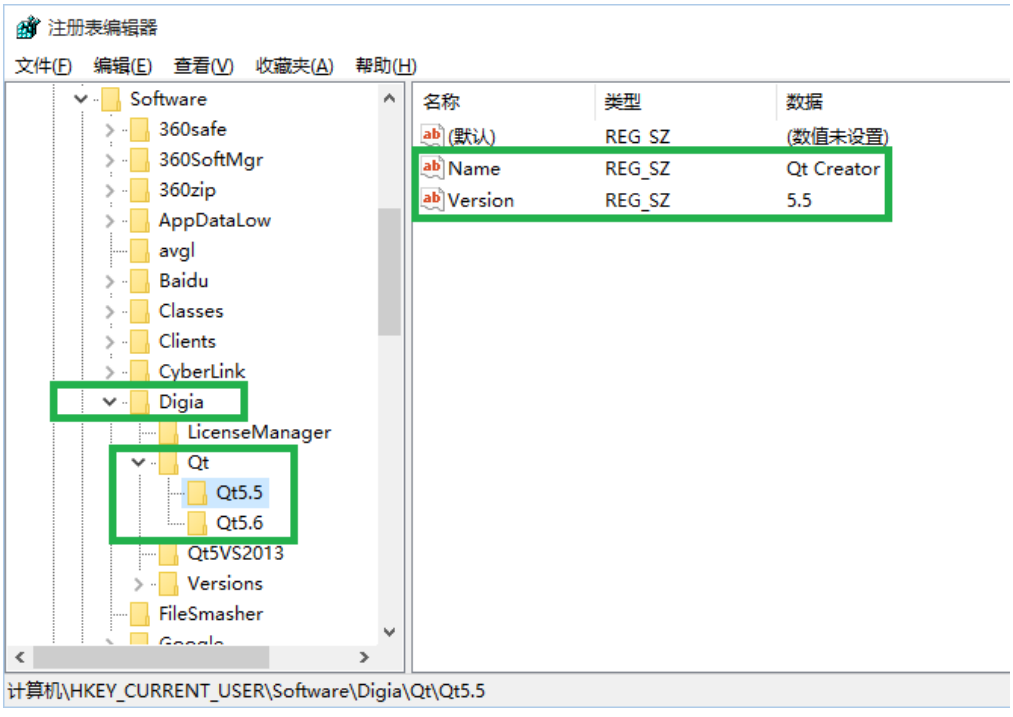
分组

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

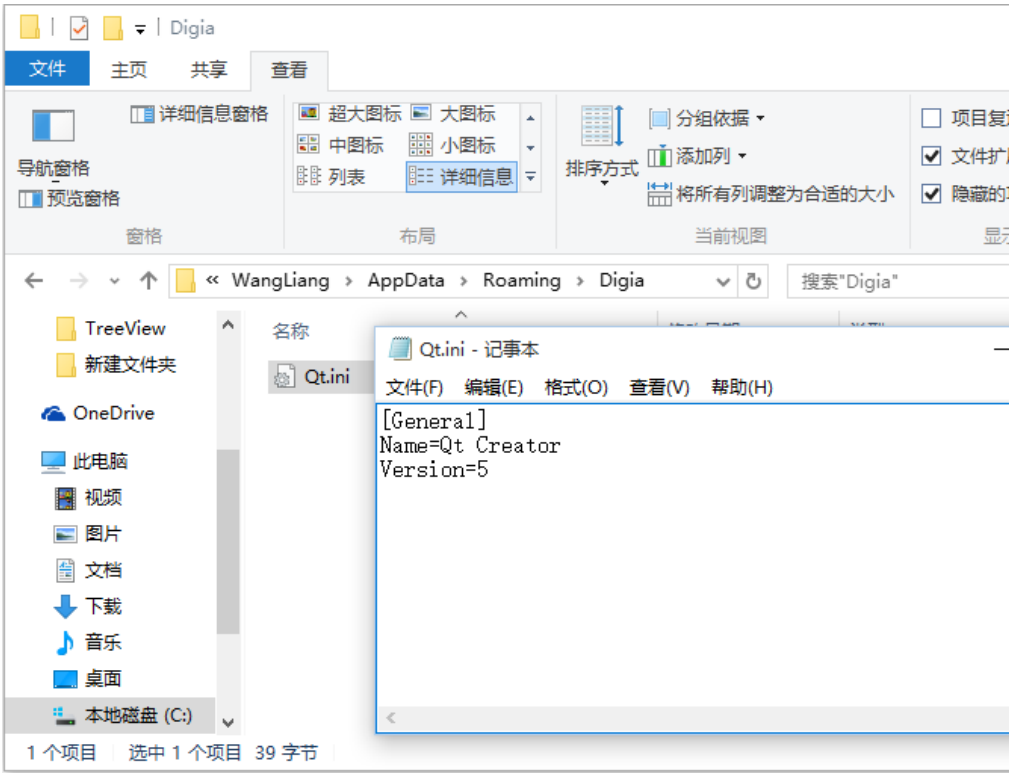
一般存储

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
```

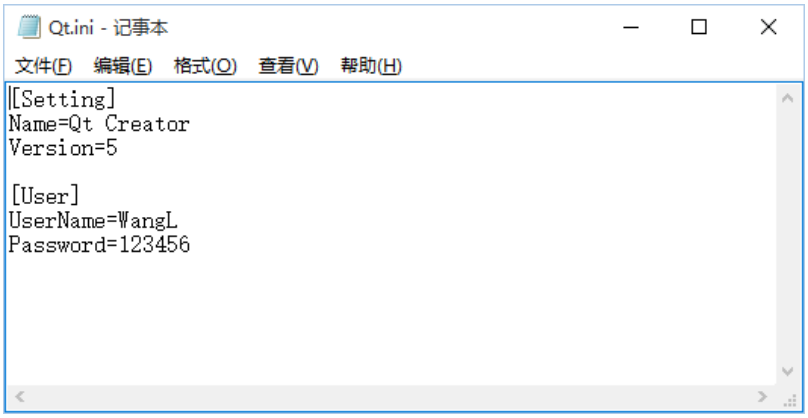
这时。我们能够通过查看应用程序输出窗体得到输出结果。

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
```

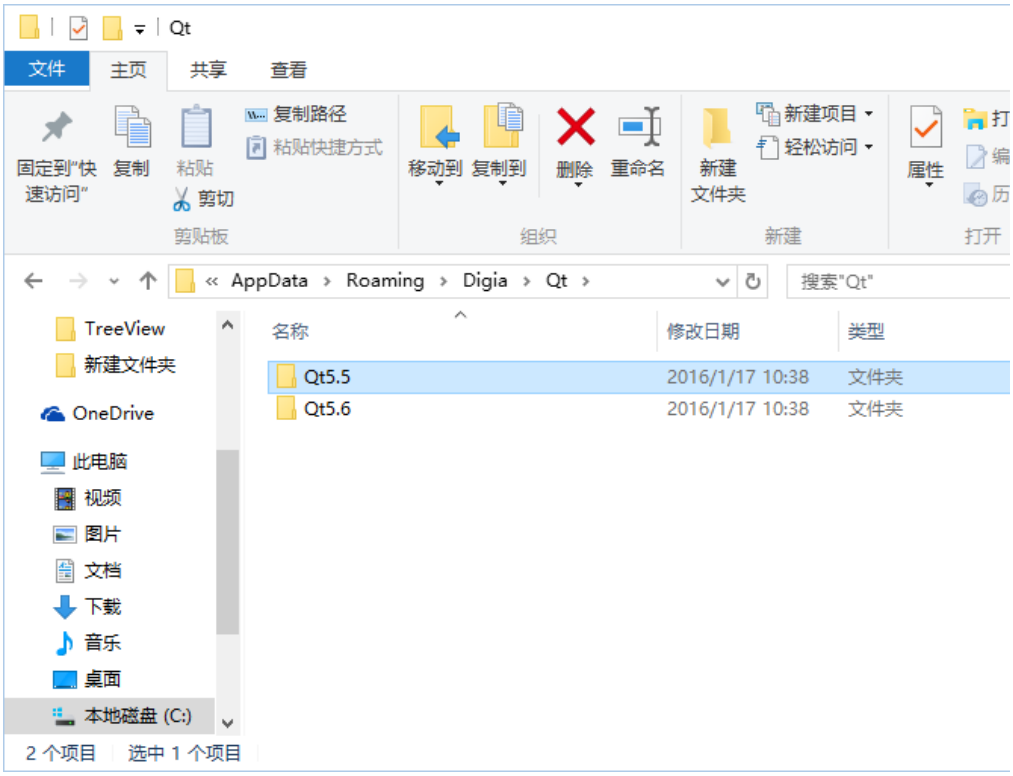
分文件夹存储

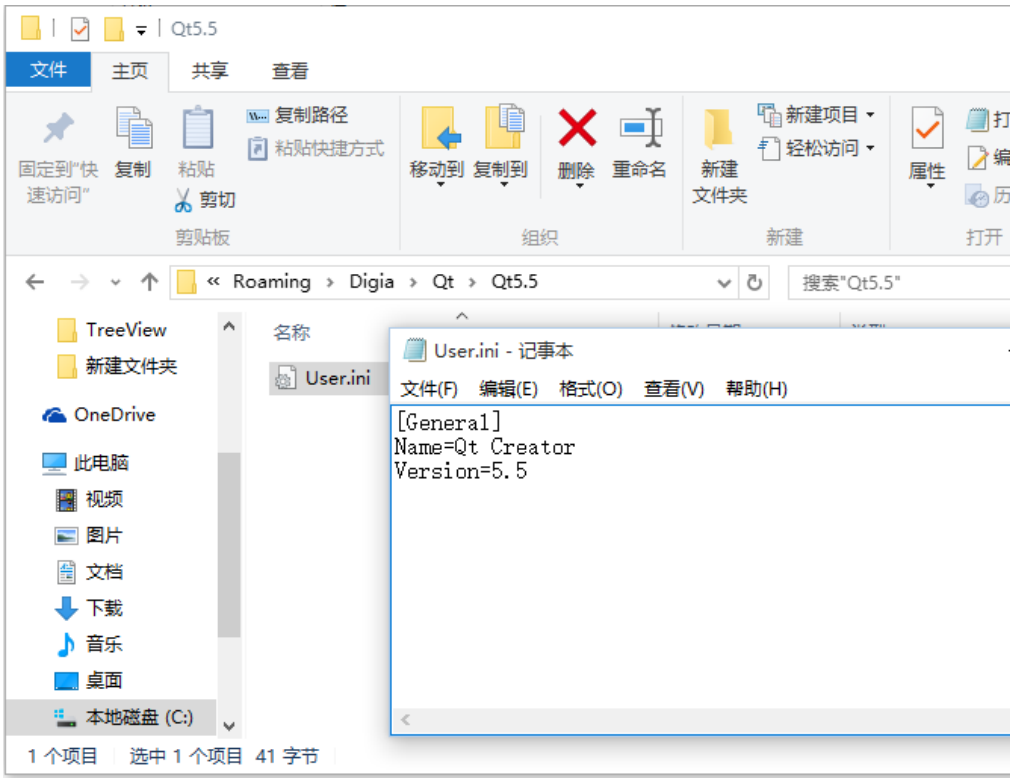
什么时候须要分文件夹存储呢？QQ大家都用过吧，是不是每个用户都有一个相应QQ号的文件夹呢，里面保存各个用户相应的信息。

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

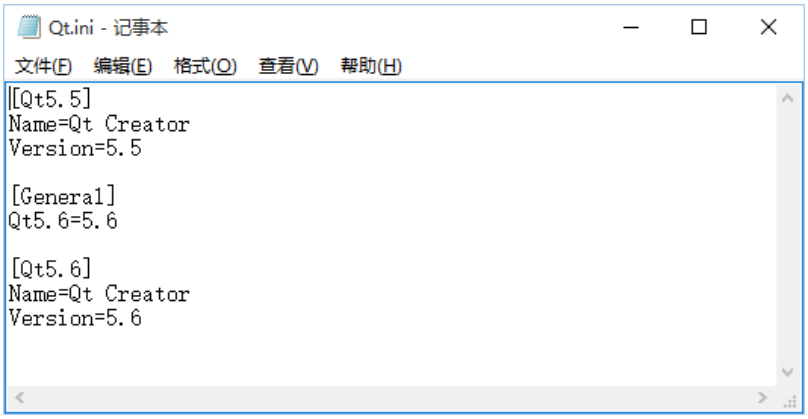
```
QSettings settings;
settings.setValue("Qt5.6", "5.6");

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。



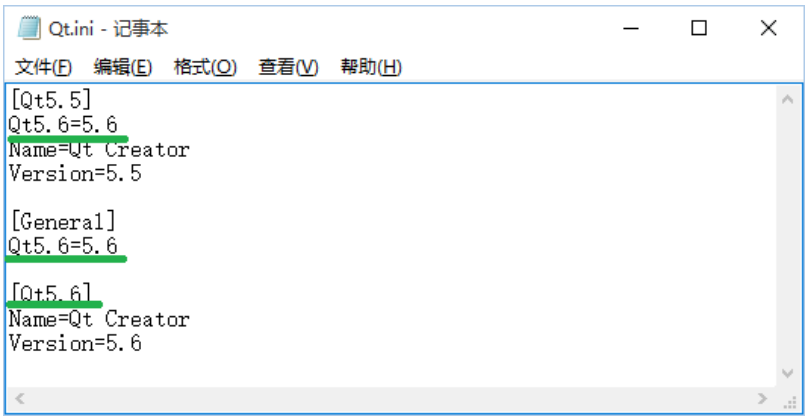
那么调用以下代码：

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

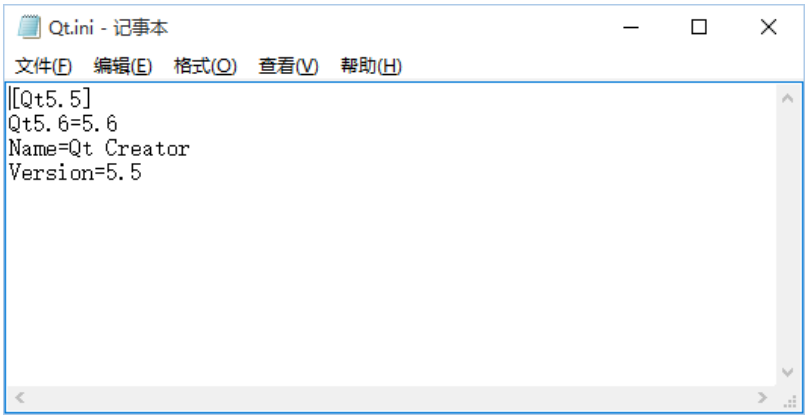
应该仅仅删除相应组中的全部键才对。也就是说剩余的keys应该为 `["Qt5.6", "Qt5.5/Name", "Qt5.5/Version"]`，General下的键为什么会被删除呢？好，这里先跳过，继续。

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。

既然这样，那么我们的疑问也就不复存在了。

很多其它参考

- QSettings介绍

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。

· 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...