C Builder注册表编程实例详解 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/253/2021_2022_C__Builder_c97_253813.htm 一、注册表编程详解 Windows 注册表中包含了系统配置、机器硬件配置、 Win32 应用程序和用户的其他配置信息。许多高级一些的功能都要通过对注册表的操作来实现。 在 WinAPI 中提供了 RegCgreateKey() 、 RegOpenKey() 、RegQueryValue() 等函数操作注册表，但是用这些函数来操作注册表使用起来非常麻烦。而利用 C ＋＋ Builder 的 TREGISTRY 类，我们则可以轻松实现对注册表的操作。下面我先介绍一下 TRegistry 类使用方法。 1. 使用前的准备工作：首先必须在程序开始处包含定义类模块的头文件： #include -registry.hpp- 其次在全程变量（即所有函数之前）创建类的一个实例（对象）： TRegistry * 实例名 =new TRegistry(). 注意：我们不能用直接声明的方法生成 TREGISTRY 的实例，这与 VC ＋＋中用 HKEY 直接生成实例的方法不同。必须采用 new 关键字生成 TREGISTRY 类的实例，然后将指针传递给声名的变量。采用这个方式声明后，实例的 RootKey 属性指向 HKEY_CURRENT_USER 根键，即默认操作是针对 HKEY_CURRENT_USER 进行的。 2. 常用属性和方法介绍：(1) 当前根键属性（ RootKey ）： RootKey 属性定义了注册表类实例当前根键，默认的是 HKEY_LOCAL_USER ， 如果我们需要在其它根键下对注册表进行操作，可以修改 RootKey 属性： MyReg->RootKey= 根键名 . BCB 中对注册表根键共有以下几个定义： HKEY_CURRENT_USER 、HKEY_CLASSES_ROOT 、 HKEY_LOCAL_MACHINE 、

HKEY_USERS、 HKEY_CURRENT_CONFIG、 HKEY_DYN_DATA、 HKEY_USERS。分别对应注册表中相应的主键。（该属性为 int 型） (2)当前键值的文本描述属性（ CurrentPath ）： CurrentPath 属性定义了注册表当前键值的文本描述，如 \HKEY_CURRENT_USER\Software\Borland 的 CurrentPath=-Software\Borland- ，而 RootKey=HKEY_CURRENT_USER （该属性为 AnsiString 型）。 (3)是否使修改后的值立即反映到注册表中（ LazyWrite ）：这个属性的作用是决定是否在执行写操作之后立即将所做的改动反映到实际的注册表中。这个属性的值在注册表对象构造时初始化为 true ，即不立即将所做的改动反映到实际的注册表中，而是在执行 CloseKey() 函数之后重写注册表，这样可以提高系统性能。但是，如果我们需要将修改立即反映到注册表（这在许多场合是必要的），则应当首先将 LazyWrite 属性设置为 false ，然后执行修改操作。 (4) 建立主键函数： bool __fastcall CreateKey(主键名). 如果主键已经存在，就覆盖原主键。如在当前主键下建立“ MyReg ”主键，可用“ CreateKey(-MyReg-) ”，而“ CreateKey(-\\MyReg-) ”则在当前根键下建立主键 MyKey 。 (5) 删除主键函数： bool __fastcall DeleteKey(主键名). 如果参数为空字符串，则删除当前键值 。 (6) 打开主键函数 : bool __fastcall OpenKey(主键名, 参数). 此函数将定位到一个具体的主键位置，随后的操作（建立键值、删除键值以及在当前位置建立主键、删除主键）将以此主键为当前主键。参数为 True 则当主键不存在将建立这个主键，如果为 false 则不建立主键。 (7) 读取当前主键下 String 型的键值函数： AnsiString __fastcall ReadString(键值名).

如： Edit1->Text=MyReg->ReadString(-MyString-). 将读取键值 MyString 的内容到文本框 Edit1 中。 同此函数类似的还有 ReadBool()、 ReadInteger()、 ReadFloat()、 ReadDateTome()、 ReadBinaryData() 等，用来读出不同类型的键值。 (8) 在当前主键中写入 String 型键值函数： void __fastcall WriteString(键值名，数据). 如果是一个新键值名，那么相当于新建一个键值；如果是已有的键值，那么就是修改键值的数据。 如： WriteString(- 我的串 -,- 内容 -). 其它类型的键值（二进制值、Dword 值）的读取和写入函数如 WriteInteger、 WriteBool()、 WriteFloat()、 WriteDateTome()、 WriteBinaryData() 等用法与上述类似。 (9) 判断键值或主键是否存在的函数： bool __fastcall ValueExists(键值名). 这个方法判断当前键下是否存在指定的数据项，如果存在返回 true，否则返回 false。 bool __fastcall KeyExists(主键名). 这个方法判断一个键是否存在，如果存在返回 true，否则返回 false。 (10)从文件读入键值函数： LoadKey(键值名 , 文件名) . (11)一个键值保存到文件函数： SaveKey(键值名 , 文件名) . (12) 关闭键值函数： void __fastcall CloseKey(void). 在注册表使用完毕后，应当及时调用 CloseKey() 成员函数关闭注册表，并调用 0delete 方法将用 new 申请的内存空间释放。 (13)当前主键下子键值的获取函数： void __fastcall GetKeyNames(Classes::TStrings * Strings) ；我们可以用该成员函数得到当前主键下所有子键的名称，用 GetKeyInfo 得到更加详细的信息。必须指出，虽然 GetKeyNames() 的说明成 void __fastcall GetKeyNames(Classes::TStrings * Strings)，也就是说，它的参数类型是 TString，但是我们并不能首先声明一个 TString 类的实

例，然后将它作为参数用于 GetKeyNames() 。这主要是由于 TStrings 类含有抽象成分。我们的解决方法是采用 TStrings 类的派生类 TStringList 来代替 TStrings 声明一个实例，并作为参数用于 GetKeyNames() 函数。 在获得子键的名称后，我们就可以利用有关函数进一步确定详细信息。如用我们可以用 GetValueNames() 结合 Read() 和 Write() 获得主键的值的详细信息。请看下面实例，这个例子的功能是将 “ \Software\MyInfo ” 主键下的所有子键名称显示在 ComboBox1 中： # include …… Tregistry * curReg=new TRegistry （） . curReg － >OpenKey(“ Software\\MyInfo-,true). KeyNames=new TStringList().// 注意 TstirngList 类的声明方法！ curReg － >GetKeyNames(KeyNames). for(int i=0.iCount.i ＋＋) ComboBox1 － >Items － >Add(KeyNames － >Strings[i]). curReg － >CloseKey(). 0delete KeyNames. 3 使用 TRegistry 的一般步骤 一般来说，有以下四步操作： 1) 建立 TRegistry 类。 2) 利用 OpenKey() 方法打开一个键值。 3) 用 ReadType() 和 WriteType() 读写键值。 4) 调用 CloseKey （） 关闭一个键值，最后调用 0delete 方法将用 new 申请的内存空间释放。二、应用实例1 下面我们通过一个示例程序演示了对注册表的常见操作，包括打开主键、读取不同类型的键值、删除键值或主键等。 “ 每次启动电脑自动运行 ” 复选框则实现的作用类似 Win 95 的 Welcome.exe 程序的功能。