C++动态创建类的实例

写在前面：首先声明，C++实际上是不可以动态创建类的实例的。

下面简单做一个解释，所谓动态创建类的实例是指在程序运行过程中创建并使用一个“未知”的类。而“未知”是指在程序编译时并不知道有哪些类是需要动态创建的。对于C++这门语言来说，编译时不知道的类是不可以在运行时使用的。所以我说C++是不可以的。

不过C++可以做到另一件事情，基本可以满足大多数类似的需求。

我描述为通过类名称创建类的实例。

进入正题。

首先描述一下需求：

编写一个程序实现在程序运行过程中通过类名称（即字符串变量）创建该类的实例，且在主程序中不出现该类的声明。

然后进行分析设计：

1.在主程序中不出现类的声明，还需要用这个类。那很自然就会想到用基类指针实现。

2.创建一个指向该类的基类指针，则需要使用new运算符。

3.通过类名调用相应的new运算符，则需要使用map加函数指针。

4.最后，如何在主函数执行之前构造该map，则需要用到静态成员在主函数运行前进行创建的机制。

下面给出代码实现：

首先有一个CObject类作为基类，由它派生出CObjectA和CObjectB两个需要动态创建的子类。

Object.h

```
#ifndef __C_OBJECT_H_
#define __C_OBJECT_H_

#include "ObjectFactory.h"

class CObject
{
public:
    CObject(): className("CObject") {}
    virtual ~CObject(){}
    virtual const std::string GetClassName()
    {
        return className;
    }
private:
    std::string className;
};

#endif //__C_OBJECT_H_
```

ObjectA.h

```
#ifndef __C_OBJECT_A_H_
#define __C_OBJECT_A_H_

#include "Object.h"

class CObjectA : public CObject
{
public:
    CObjectA(): className("CObjectA") {}
    ~CObjectA(){}
    const std::string GetClassName()
    {
        return className;
    }
private:
    std::string className;
};

REGISTER_CLASS(CObjectA);

#endif //__C_OBJECT_A_H_
```

ObjectB.h

```
#ifndef __C_OBJECT_B_H_
#define __C_OBJECT_B_H_

#include "Object.h"

class CObjectB : public CObject
{
public:
    CObjectB(): className("CObjectB") {}
    ~CObjectB(){}
    const std::string GetClassName()
    {
        return className;
    }
private:
    std::string className;
};

REGISTER_CLASS(CObjectB);

#endif //__C_OBJECT_B_H_
```

然后重点就来了，上面两个类里面都有的宏定义REGISTER_CLASS。实际上就是声明一个带有静态成员的注册类，通过初始化该静态成员实现构造动态创建map。

ObjectFactory.h

```
#ifndef __C_OBJECT_FACTORY_H_
#define __C_OBJECT_FACTORY_H_

#include <map>
#include <string>

typedef void* (*NewInstancePt)();

class CObjectFactory
{
public:
    static void* CreateObject(const char *className)
    {
        std::map<std::string, NewInstancePt>::const_iterator it;
        it = dynCreateMap.find(className);
        if(it == dynCreateMap.end())
            return NULL;
        else
        {
            NewInstancePt np = it->second;
            return np();
        }
    }

    static void RegisterClass(const char *className, NewInstancePt np)
    {
        dynCreateMap[className] = np;
    }
private:
    static std::map<std::string, NewInstancePt> dynCreateMap;
};

std::map<std::string, NewInstancePt> CObjectFactory::dynCreateMap;

class Register
{
public:
    Register(const char *className, NewInstancePt np)
    {
        CObjectFactory::RegisterClass(className, np);
    }
};

#define REGISTER_CLASS(class_name) \
class class_name##Register \
{ \
public: \
    static void* NewInstance() \
    { \
        return new class_name(); \
    } \
private: \
    static Register reg; \
}; \
Register class_name##Register::reg(#class_name, class_name##Register::NewInstance)

#endif //__C_OBJECT_FACTORY_H_
```

最后，当然还有如何使用的主函数内容。

RTTI.cpp

```
#include <stdio.h>
#include "ObjectFactory.h"
#include "Object.h"
#include "ObjectA.h"
#include "ObjectB.h"

int main(int argc, const char *argv[])
{
    CObject *objA = static_cast<CObject *>(CObjectFactory::CreateObject("CObjectA"));
    std::string className;
    if(objA == NULL)
    {
        printf("[ERROR] Can't Create Class ObjectA!\n");
    }
    else
    {
        className = objA->GetClassName();
        printf("[OK] Create %s !\n", className.c_str());
    }

    CObject *objB = static_cast<CObject *>(CObjectFactory::CreateObject("CObjectB"));
    if(objB == NULL)
    {
        printf("[ERROR] Can't Create Class ObjectB!\n");
    }
    else
    {
        className = objB->GetClassName();
        printf("[OK] Create %s !\n", className.c_str());
    }

    return 0;
}
```

这样就完成了所谓的动态创建。这里需要注意的是在主函数所在的文件里面需要include所有的需要动态创建的类，否则的话在编译的过程中这些类以及其注册类就根本不会被编译，也就不会构建动态创建map，整个机制也就失效了。这也就是我说的C++实际上是不可以动态创建类的原因。

不得不说，如果想要彻底的动态创建，建议使用Java。简单看看Java的反射机制和ClassLoader就会知道原来动态加载如此容易。。。

标签: C++, RTTI, OOP

好文要顶 关注我 收藏该文

asworld
关注 - 0
粉丝 - 0
+加关注

2 推荐 0 反对

« 上一篇： Linux下非root权限安装与使用GDAL库的方法

asworld
博客园 首页 新随笔 联系 管理 订阅
昵称： asworld
园龄： 7年11个月
粉丝： 0
关注： 0
+加关注

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
更多链接

我的标签
C++(2)
OOP(1)
RTTI(1)
Linux(1)
GDAL(1)
Centos(1)
ImageMagick(1)
JMagick(1)

随笔档案
2014年10月(2)
2014年8月(1)

阅读排行榜
1. C++动态创建类的实例(9150)
2. Linux下非root权限安装与使用GDAL库的方法(2481)
3. Centos 6 安装 JMagick(5244)

评论排行榜
1. C++动态创建类的实例(4)

推荐排行榜
1. C++动态创建类的实例(2)

最新评论
1. Re:C++动态创建类的实例
我自己也研究了一段时间，发现需要动态创建的类必须最终在主函数所在的项目中编译才有效，如果在外部的动态库中编译，再把动态库链接到主程序就没效果。
--duxin_kaisa
2. Re:C++动态创建类的实例
不错，刚好用得着
--sumejing
3. Re:C++动态创建类的实例
工厂方法。。。
--eryar
4. Re:C++动态创建类的实例
工厂模式的一种变形
--zangyu00544