Subject: Re: Template class factory
Posted by mdelfede on Wed, 13 Jan 2010 14:37:28 GMT
View Forum Message <> Reply to Message

Well, quite interesting topic

The definitive factory, thanx to Mirek know-how :

```
#ifndef _ClassFactory_ClassFactory_h_
#define _ClassFactory_ClassFactory_h_

#include <Core/Core.h>
using namespace Upp;

template<class T> class WithFactory
{
 private:
  typedef One<T> (*CreateFunc)();
  typedef VectorMap<String, CreateFunc>mapType;
  static mapType &classMap() { static mapType cMap; return cMap; }
  static VectorMap<String, String> &typeMap() { static VectorMap<String, String> tMap; return
tMap; }
  template<class D> static One<T> __Create(void) { return One<T>((T *)new D); }
 public:

  template<class D> static void Register(const String &name)
  {
   classMap().Add(name, __Create<D>);
   typeMap().Add(typeid(D).name(), name);
  }
  static One<T> Create(const String &className) { return classMap().Get(className)(); }
  static T *CreatePtr(String const &className) { return classMap().Get(className)().Detach(); }
  static Vector<String> const &Classes(void) { return classMap().GetKeys(); }
  String const &IsA(void) { return typeMap().Get(typeid(*this).name()); }
  virtual ~WithFactory() {}
};

#define REGISTERCLASS(type) \
 INITBLOCK { \
 type::Register<type>(#type); \
}
#endif
```

and the sample usage :

```
#include "ClassFactory.h"
```

```
class Base : public WithFactory<Base>
{
};

class Derived : public Base
{
};

REGISTERCLASS(Base);
REGISTERCLASS(Derived);

CONSOLE_APP_MAIN
{
 One<Base> p1 = Base::Create("Base");
 One<Base> p2 = Base::Create("Derived");

 Cerr() << "p1 is a '" << p1->IsA() << "'\n";
 Cerr() << "p2 is a '" << p2->IsA() << "'\n";

 Base *ptr = Base::CreatePtr("Derived");
 Cerr() << "ptr  is a '" << ptr->IsA() << "'\n";
 delete ptr;

 Base *d = new Derived;
 Cerr() << "d  is a '" << d->IsA() << "'\n";
 delete d;

}
```