Subject: Re: Template class factory
Posted by mdelfede on Wed, 13 Jan 2010 10:40:59 GMT
View Forum Message <> Reply to Message

A better version with RTTI class identification :

```
#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; }
  template<class D> static One<T> __Create(void) { return One<T>((T *)new D); }
  String myType;
 public:

  template<class D> static void Register(const String &name) { classMap().Add(name,
__Create<D>); }
  static One<T> Create(const String &className)
  {
   One<T> me = classMap().Get(className)();
   me->myType = className;
   return me;
  }
  static Vector<String> const &Classes(void) { return classMap().GetKeys(); }
  String const &IsA(void) { return myType; }
};

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

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";

}
```

Now classes knows themselves

Ciao

Max
