Subject: Template class factory
Posted by mdelfede on Wed, 13 Jan 2010 10:17:46 GMT
View Forum Message <> Reply to Message

Having the need to create some class instances at runtime by class name (String, not type), I developed a small templatized class factory; it's the first step for the polymorphic xmlizer that I'll publish when ready.

Thank to Mirek who helped me with templates !

ClassFactory.h

```
#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); }
 public:

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

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

Sample usage:

```
#include "ClassFactory.h"

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

```
  virtual String WhoAmI(void) { return "Base"; }
};

class Derived : public Base
{
 public:
  virtual String WhoAmI(void) { return "Derived"; }
};

REGISTERCLASS(Base);
REGISTERCLASS(Derived);

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

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

}
```

output :

```
p1 is a 'Base'
p2 is a 'Derived'
```

Ciao

Max