Subject: Re: What about LUA plugin?
Posted by mirek on Sun, 10 Sep 2006 22:24:34 GMT
View Forum Message <> Reply to Message

Quote:
```
struct XmlIO {
  template<class T>  XmlIO& operator()(const char* tag, T& v)
    { std::cout << "<" << tag << ">" << v << "<\\" << tag << ">"; return *this; }
};
```

It is bidirectional, so cout does not make it, however I see the point...

Quote:
```
#define SERIALIZABLE(Class)  template<class stream> friend stream& operator<<(stream&,
Class&);

// End user code
class C {
  int v1, v2;
  SERIALIZABLE(Class);
public:
  C() : v1(12), v2(24) {}
};
```

Yep. But using macro is ugly and requires more typing. Instead, there is

```
template <class T> XmlIO XmlIO::operator()(const char *tag, T& var) {
 XmlIO n(*this, tag);
 Xmlize(n, var);
 return *this;
}
```

soo if you want to be aspect aware, which I understand as being able to add xmlization to existing class, simply define Xmlize *function* for it.

Xmlize itself has template

```
template <class T>
void Xmlize(XmlIO xml, T& var)
{
 var.Xmlize(xml);
}
```

(Now if somebody asks what we mean by "aggresive use of C++", here is the answer).

Note that sometimes you might want to define more funtions to xmlize single type to express the value the type really represents, e.g. there is XmlizeLang that interprets int as language indentifier (e.g. "en-us").

Quote:
About a uppforge, why waiting high demand? Won't the offer create demand? At least, this shall only encourage sharing U++ code, thus giving more examples of U++ code. Maybe just letting a space where to put code snipset would be nice.

What I really wanted to say is that U++ is not popular enough for somebody to start one. If you feel like starting such effort, go for it, it will definitely help us.

Mirek
