Subject: Re: What about LUA plugin?
Posted by thierry on Sun, 10 Sep 2006 21:47:38 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 10 September 2006 20:50thierry wrote on Sun, 10 September 2006 13:45Ok. And you shall add Esc for custom controls in layout editor.

No, I will not add Esc for custom controls. I have done so 2 years ago

Quote:
Then I just wonder if there would be kind a uppforge.net for custom packages !

I hope that in future, yes. At the moment, we are not as popular yet

Mirek

About Esc, I wasn't saying the intregation of Esc was a todo, but just mentionning it as yet another configuration mean of theIDE to add to the list xml, serialize.

I was more trying to put the emphasis on the fact that configuration means were multiplying. All with their pros and cons, then without deep design consideration about why using them other than marketing done around them.

But in the end they all tend to be used, thus asking more code to be written (even if small, this is still a bloat) and cores to be linked.

Given machine strength, most of the needs could be satisfied with one small core script language, like lua proves, but why not another as long as it can have a critical mass, or a database when data are larger.

Then why paying the effort of maintaining a script language where you don't want to maintain a database engine? And to which extent, this can predate the development of other nice features.

Don't take it for U++, this is more general consideration about users architecture decision, and a tool like U++ aims to address as many kind of users needs as possible. Including yours at first with your existent code base.

My only worry would be that serialization is more supported by external streaming operators in a more modular and extensible aspect oriented design.
something like:

```
#include <iostream>

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

#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) {}
};

template<> std::ostream& operator<<(std::ostream& s, C& c) { return s << c.v1 << std::endl <<
c.v2; }
template<> XmlIO& operator<<(XmlIO& xml, C& c)   { return xml("v1", c.v1)("v2", c.v2); }

int main() {
  C c;
  std::cout << c << std::endl;
  XmlIO s;
  s << c;
  return 0;
}
```

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.