Subject: Xmlize works only for storing
Posted by exhu on Wed, 01 Oct 2008 13:16:33 GMT
View Forum Message <> Reply to Message

Please, help understanding the XML serialization implemented in UPP.

This code perfectly saves the XML as intended, but does not load attribute values:

```
void ServData::save() {
   StoreAsXMLFile(*this);
}

void ServData::load() {
   LoadFromXMLFile(*this);
}

void ServData::Xmlize(XmlIO xml) {
   ::Xmlize(xml.Add("map"), locationMap);
}

/////////


void Xmlize(XmlIO xml, Map & locMap) {

   Xmlize(xml.Add("left"), locMap.left);
   Xmlize(xml.Add("right"), locMap.right);
   Xmlize(xml.Add("top"), locMap.top);
   Xmlize(xml.Add("bottom"), locMap.bottom);
}

void Xmlize(XmlIO xml, MapPlace & place) {
   String nm;

 if (xml.IsStoring())
  nm = place.name;


   xml.Attr("name", nm); // nm is always empty on xml.IsLoading! why?

  if (xml.IsLoading())
   place.name = nm;

}
```

Subject: Re: Xmlize works only for storing
Posted by Mindtraveller on Wed, 01 Oct 2008 20:27:44 GMT
View Forum Message <> Reply to Message

I doubt if
Xmlize(xml.Add("***"), ***);
would work in both directions.

Why don`t you use construction from Xmlize reference sample:

```
void *****::*****::Xmlize(XmlIO xml)
{
 xml
  ("***", ***)
  ("***", ***)
 ;
}
```

Subject: Re: Xmlize works only for storing
Posted by exhu on Thu, 02 Oct 2008 07:08:55 GMT
View Forum Message <> Reply to Message

Quote:Why don`t you use construction from Xmlize reference sample:

I can't use it because STL and other simple types already defined in the program do not contain Xmlize methods which are called by the template.

Changed to:

```
void ServData::Xmlize(XmlIO xml) {
    ::Xmlize(XmlIO(xml,"map"), locationMap);
    //XmlIO(xml, "magic").Attr("magic", magic);
    //magic = magic;
}

/////////


void Xmlize(XmlIO xml, Map & locMap) {
    xml.Attr("shopname", locMap.shopName);
    Xmlize(XmlIO(xml, "left"), locMap.left);
    Xmlize(XmlIO(xml,"right"), locMap.right);
    Xmlize(XmlIO(xml,"top"), locMap.top);
    Xmlize(XmlIO(xml,"bottom"), locMap.bottom);

}
```

Now it works, but it's not obvious why because both Add() method and XmlIO() constructor use & (reference) for variable argument.

Who can explain this magic? No comments at all in the library sources

---

Subject: Re: Xmlize works only for storing
Posted by Mindtraveller on Thu, 02 Oct 2008 17:22:50 GMT
View Forum Message <> Reply to Message

[quote title=exhu wrote on Thu, 02 October 2008 11:08]Quote:I can't use it because STL and other simple types already defined in the program do not contain Xmlize methods which are called by the template.
Sorry I can`t clearly understand what are you talking about.

Subject: Re: Xmlize works only for storing
Posted by exhu on Fri, 03 Oct 2008 06:37:31 GMT
View Forum Message <> Reply to Message

Try compiling,

std::list<MyType> mylist;

XmlIO(xml, mylist);

And you'll get errors like "T.Xmlize: The class does not define a method Xmlize"...

Subject: Re: Xmlize works only for storing
Posted by Mindtraveller on Fri, 03 Oct 2008 07:13:49 GMT
View Forum Message <> Reply to Message

I see no critical problem with this. You may write your own class derived from std::list<...> with Xmlize function.
Besides I do not thnk it is good idea to mix STL and NTL libraries in code.

Subject: Re: Xmlize works only for storing
Posted by mirek on Fri, 03 Oct 2008 09:54:28 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Fri, 03 October 2008 03:13I see no critical problem with this. You may write

your own class derived from std::list<...> with Xmlize function.
Besides I do not thnk it is good idea to mix STL and NTL libraries in code.

Well, I certainly would not recomend using STL but sometimes you perhaps need to deal with existing code...

You can define Xmlize as template function specialisation and that solves the problem of "external" types:

```
#include <Core/Core.h>
#include <vector>

using namespace Upp;
using namespace std;

template<> void Upp::Xmlize(XmlIO xml, vector<int>& data) {
 if(xml.IsStoring())
  for(int i = 0; i < (int)data.size(); i++)
   Xmlize(xml.Add("item"), data[i]);
 else {
  data.clear();
  for(int i = 0; i < xml->GetCount(); i++)
   if(xml->Node(i).IsTag("item")) {
    data.push_back(0);
    Xmlize(xml.At(i), data.back());
   }
 }
}

CONSOLE_APP_MAIN
{
 vector<int> x;
 x.push_back(1);
 x.push_back(2);
 x.push_back(3);
 String s = StoreAsXML(x, "std-test");
 DUMP(s);
 vector<int> y;
 LoadFromXML(y, s);
 for(int i = 0; i < (int)y.size(); i++)
  DUMP(y[i]);
}
```

Mirek

Subject: Re: Xmlize works only for storing
Posted by mirek on Fri, 03 Oct 2008 09:57:33 GMT
View Forum Message <> Reply to Message

PS.: added to reference examples...

Subject: Re: Xmlize works only for storing
Posted by exhu on Mon, 06 Oct 2008 07:45:36 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 03 October 2008 12:54

Well, I certainly would not recomend using STL

Ok, but are there analogues in NTL to the following STL classes:
set, list ?

Subject: Re: Xmlize works only for storing
Posted by mirek on Tue, 07 Oct 2008 10:41:02 GMT
View Forum Message <> Reply to Message

exhu wrote on Mon, 06 October 2008 03:45luzr wrote on Fri, 03 October 2008 12:54

Well, I certainly would not recomend using STL

Ok, but are there analogues in NTL to the following STL classes:
set, list ?

std::set -> Index. It provides something a bit more complex, but can easily replace set and multiset.

std::list is simply completely useless container. Use Vector/Array/BiVector/BiArray.

(Before you start argumenting about O(1) insertion times, tell how do you know where to insert

Mirek