Subject: Re: ValueArray behaviour / inconsistantcy / BUG?
Posted by kohait00 on Fri, 22 Oct 2010 11:52:35 GMT
View Forum Message <> Reply to Message

ValueMap is sure more complicated. i tried it..realizing btw. that Index and ArrayIndex are not Xmlize'able yet. because they dont export API to access the hash separetely, so one can't cleanly xmlize it, only under assumption that hashes have been created from the objects them selves; nevertheless, code explains more.

and another bug (dunno, could be): XmlizeMap does not Clear() the container at the beginning, like XmlizeContainer does.

NAMESPACE_UPP

```
//old version without preparaton for hashfn awareness
#if 0
template<class T>
void Xmlize(XmlIO xml, Index<T>& data)
{
 if(xml.IsStoring())
 {
  XmlizeStore(xml, data.GetKeys()); //FIXME xmlize with hashfn awareness
 }
 if(xml.IsLoading())
 {
  data.Clear();
  Vector<T> keys;
  ::Xmlize(xml, keys); //FIXME dexmlize with hashfn awareness
  data = Index<T>(keys);
 }
}
#endif

template<class K, class T>
void XmlizeIndex(XmlIO xml, const char *keytag, const char *valuetag, T& data)
{
 if(xml.IsStoring()) {
  for(int i = 0; i < data.GetCount(); i++)
   if(!data.IsUnlinked(i)) {
    //XmlizeStore(xml.Add(keytag), data.GetKey(i)); //FIXME xmlize with hashfn awareness
    XmlizeStore(xml.Add(valuetag), data[i]);
   }
 }
 else {
  data.Clear();
  int i = 0;
```

```
  //while(i < xml->GetCount() - 1 && xml->Node(i).IsTag(keytag) && xml->Node(i +
1).IsTag(valuetag)) {
  while(i < xml->GetCount() && xml->Node(i).IsTag(valuetag)) {
   //K key;
   //Xmlize(xml.At(i++), key); //FIXME dexmlize with hashfn awareness
   K k;
   ::Xmlize(xml.At(i++), data.Add(k));
  }
 }
}

template<class K, class H>
void Xmlize(XmlIO xml, Index<K, H>& data)
{
 XmlizeIndex<K>(xml, "key", "value", data);
}

template<class K, class H>
void Xmlize(XmlIO xml, ArrayIndex<K, H>& data)
{
 XmlizeIndex<K>(xml, "key", "value", data);
}

template<> void Xmlize(XmlIO xml, ValueArray& v)
{
 if(xml.IsStoring())
 {
  XmlizeStore(xml, v.Get());
 }
 if(xml.IsLoading())
 {
  Vector<Value> vv;
  ::Xmlize(xml, vv);
  v = ValueArray(vv);
 }
}

template<> void Xmlize(XmlIO xml, ValueMap& v)
{
 if(xml.IsStoring())
 {
  XmlizeStore(xml, v.GetKeys());
  XmlizeStore(xml, v.GetValues());
 }
 if(xml.IsLoading())
 {
  Index<Value> vv;
  ::Xmlize(xml, vv);
```

```
  ValueArray va;
  ::Xmlize(xml, va);
  ASSERT(vv.GetCount() == va.GetCount());
  ValueMap vm;
  for(int i = 0; i < vv.GetCount(); i++)
   vm.Add(vv[i], va[i]);
 }
}
END_UPP_NAMESPACE
```