Subject: Jsonize problems with maps
Posted by Mindtraveller on Wed, 18 Apr 2012 07:11:58 GMT
View Forum Message <> Reply to Message

There's a problem with maps jsonization.
Small example:#include <Core/Core.h>
using namespace Upp;

```
struct TestStruct
{
 VectorMap<String,String> map;
 void Jsonize(JsonIO &json)
 {
  json("map",map); //compile error!
 }
};

CONSOLE_APP_MAIN
{
 StoreAsJson(TestStruct());
}
```

Actually, I found 2 problems with jsonizing the Maps.
1) It doesn't actually compile (strange!)
2) Even if it compiled, json representation of such map is not optimal. IMO there must be special version for maps with string keys. It will make possible using native json way for representing such maps:
{ "key1":value1, "key2":value2, ... }

Subject: Re: Jsonize problems with maps
Posted by mirek on Wed, 18 Apr 2012 07:34:59 GMT
View Forum Message <> Reply to Message

1) solved

2) well, I am not 100% happy about it too, but I guess what we have is still the most reasonable solution right now. I am afraid that doing exception for String keys is not quite a good idea.

Perhaps we could settle for special version Jsonize that handles it differently?

Subject: Re: Jsonize problems with maps
Posted by Mindtraveller on Wed, 18 Apr 2012 17:15:29 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 18 April 2012 11:34Perhaps we could settle for special version Jsonize that

handles it differently?
Agree with your ideas about keeping "common" approach to maps/vectors.
Making some special version for string keyed maps is exactly what I'm talking about.
Something like
template<class V> JsonizeStringMap(JSonIO &, ArrayMap<String,V> &) {...}

Subject: Re: Jsonize problems with maps
Posted by mirek on Wed, 18 Apr 2012 17:37:48 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Wed, 18 April 2012 13:15mirek wrote on Wed, 18 April 2012 11:34Perhaps we could settle for special version Jsonize that handles it differently?
Agree with your ideas about keeping "common" approach to maps/vectors.
Making some special version for string keyed maps is exactly what I'm talking about.
Something like
template<class V> JsonizeStringMap(JSonIO &, ArrayMap<String,V> &) {...}

Will you propose a patch?

Mirek

Subject: Re: Jsonize problems with maps
Posted by Mindtraveller on Wed, 18 Apr 2012 23:03:24 GMT
View Forum Message <> Reply to Message

First, I had to implement two member functions of ValueMap class, which were declared in Documantation but were actually absent: const Value& GetKey(int i) const { return data->key[i]; }
const Value& GetValue(int i) const { return data->value[i]; }

So, here is the patch:template <class T, class K, class V>
void JsonizeStringMap(JsonIO& io, T& map)
{
if(io.IsLoading()) {
map.Clear();
const ValueMap& va = io.Get();
map.Reserve(va.GetCount());
for(int i = 0; i < va.GetCount(); i++) {
Value vv = va[i];
K key;
V value;
LoadFromJsonValue(key, va.GetKey(i));
LoadFromJsonValue(value, va.GetValue(i));
map.Add(key, value);

```
  }
 }
 else {
  Index<Value>  index;
  Vector<Value> values;
  index .Reserve(map.GetCount());
  values.Reserve(map.GetCount());
  for (int i=0; i<map.GetCount(); ++i)
  {
   index .Add(StoreAsJsonValue(map.GetKey(i)));
   values.Add(StoreAsJsonValue(map[i]));
  }
  ValueMap vm(index, values);
  io.Set(vm);
 }
}

template <class K, class V, class H>
void Jsonize(JsonIO& io, VectorMap<K, V, H>& map, bool)
{
 JsonizeStringMap<VectorMap<K, V, H>, K, V>(io, map);
}

template <class K, class V, class H>
void Jsonize(JsonIO& io, ArrayMap<K, V, H>& map, bool)
{
 JsonizeStringMap<ArrayMap<K, V, H>, K, V>(io, map);
}
```

Here is simple demo:#include <Core/Core.h>
using namespace Upp;

```
struct TestStruct
{
 struct TestV : Moveable<TestV>
 {
  int a;
  int b;
  String ToString() const
  {
   return Format("a=%d, b=%d", a,b);
  }

  void Jsonize(JsonIO &json)
  {
   json
    ("a", a)
```

```
    ("b", b)
   ;
  }
 };

 void Add()
 {
  TestV v;
  v.a = Random(100);
  v.b = Random(100);
  map.AddPick(FormatIntHex(Random() ^ (int) GetTickCount()), v);
 }

 void Jsonize(JsonIO &json)
 {
  //Upp::Jsonize(json,map); // <- default
  ::Jsonize(json,map,true); // <- string map
 }
 VectorMap<String,TestV> map;
};

CONSOLE_APP_MAIN
{
 TestStruct test, test2;
 test.Add();
 test.Add();

 LoadFromJson(test2, StoreAsJson(test));

 Cout() << StoreAsJson(test) << "\n============================\n" << StoreAsJson(test2)
<< "\n\n";
}
```

I'm not quite shure about efficiency of my implementation, but tried to comply Mirek's code as much as possible.

P.S. It would be good to have a mechanizm for "pretty" serialization. Possible interface could be: StoreAsJson(const T&, bool pretty = false). Is it possible to implement? Should I make a patch for it?
This will make possible storing program configuration in JSON format, not in XML. Which must be much more efficient and understandable.

Subject: Re: Jsonize problems with maps
Posted by mirek on Thu, 19 Apr 2012 07:20:20 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Wed, 18 April 2012 19:03First, I had to implement two member functions of

ValueMap class, which were declared in Documantation but were actually absent: const Value& GetKey(int i) const { return data->key[i]; }
const Value& GetValue(int i) const { return data->value[i]; }

Just for explanation:

They are not present in "legacy" Value implementation. Documentation is about new "SVO Value".

I still did not had balls to finally switch SVO Value as default (it was delayed as I have found a subtle bug a month ago)... but it is definitely coming before the end of May.

So, here is the patch:template <class T, class K, class V>
void JsonizeStringMap(JsonIO& io, T& map)
{
if(io.IsLoading()) {
map.Clear();
const ValueMap& va = io.Get();
map.Reserve(va.GetCount());
for(int i = 0; i < va.GetCount(); i++) {
Value vv = va[i];
K key;
V value;
LoadFromJsonValue(key, va.GetKey(i));
LoadFromJsonValue(value, va.GetValue(i));
map.Add(key, value);
}
}
else {
Index<Value> index;
Vector<Value> values;
index .Reserve(map.GetCount());
values.Reserve(map.GetCount());
for (int i=0; i<map.GetCount(); ++i)
{
index .Add(StoreAsJsonValue(map.GetKey(i)));
values.Add(StoreAsJsonValue(map[i]));
}
ValueMap vm(index, values);
io.Set(vm);
}
}

template <class K, class V, class H>
void Jsonize(JsonIO& io, VectorMap<K, V, H>& map, bool)
{
JsonizeStringMap<VectorMap<K, V, H>, K, V>(io, map);

```
}

template <class K, class V, class H>
void Jsonize(JsonIO& io, ArrayMap<K, V, H>& map, bool)
{
 JsonizeStringMap<ArrayMap<K, V, H>, K, V>(io, map);
}
```

Here is simple demo:

```
#include <Core/Core.h>
using namespace Upp;

struct TestStruct
{
 struct TestV : Moveable<TestV>
 {
  int a;
  int b;
  String ToString() const
  {
   return Format("a=%d, b=%d", a,b);
  }

  void Jsonize(JsonIO &json)
  {
   json
    ("a", a)
    ("b", b)
   ;
  }
 };

 void Add()
 {
  TestV v;
  v.a = Random(100);
  v.b = Random(100);
  map.AddPick(FormatIntHex(Random() ^ (int) GetTickCount()), v);
 }

 void Jsonize(JsonIO &json)
 {
  //Upp::Jsonize(json,map); // <- default
  ::Jsonize(json,map,true); // <- string map
 }
 VectorMap<String,TestV> map;
};
```

```
CONSOLE_APP_MAIN
{
 TestStruct test, test2;
 test.Add();
 test.Add();

 LoadFromJson(test2, StoreAsJson(test));

 Cout() << StoreAsJson(test) << "\n============================\n" << StoreAsJson(test2)
<< "\n\n";
}
```
[/quote]

Applied, only I do not really see the point of Jsonize with another bool parameter, that is why I have renamed it to StringMap. Thanks!

Mirek

Subject: Re: Jsonize problems with maps
Posted by Mindtraveller on Thu, 19 Apr 2012 08:01:06 GMT
View Forum Message <> Reply to Message

Mirek, what do you think about "pretty" jsonization as option in StoreAsJson()? As I mentioned before, without it, storing some types of data (i.e. program config) in JSON would be practically impossible.
Are you agree to add this option? Should I deliver a patch for it?

Subject: Re: Jsonize problems with maps
Posted by mirek on Thu, 19 Apr 2012 09:15:19 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 19 April 2012 04:01Mirek, what do you think about "pretty" jsonization as option in StoreAsJson()? As I mentioned before, without it, storing some types of data (i.e. program config) in JSON would be practically impossible.
Are you agree to add this option? Should I deliver a patch for it?

I agree, it is now there.

Mirek
