Subject: Re: NEW: generic Toupel grouper
Posted by dolik.rce on Mon, 18 Oct 2010 09:12:48 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 18 October 2010 10:00ok i get it. this one is then possible

```
 Tuple2<int, const char *> data = { 1, "Kvuli" };
```

, though its a bit more verbose then

```
Tuple2<int, const char*>(1,"Kvuli);
```

The idea is that with Tuple2<...>(a,b) you can't fill the arrays easily, as Mirek was pointing out.

I have read more about the requirements for type to be POD. If I understand right, then the only thing preventing Tuple from being POD and Moveable same time is the inheritance from Moveable. So that leaves us with two options: Either declare the friend void AssertMoveable0(T*) directly in the Tuple struct or doing it outside with (something like) NTL_MOEABLE(). The first approach is troublesome because it is not easy to do that only for tuples with moveable contents. The second option is useless too, since outside of the template is no chance to do it automaticaly, as the template parameters are not known.

Also there might be one more possibility. Is it possible to mark class moveable in other class? If I am not mistaken the entire trick is to make the compiler instantiate the AssertMoveable0(T*). So I would expect something like this to work too: template<class A,class B>
struct helper__Tuple2<A,B>:PossiblyMoveable<Tuple2<A,B>,A,B>{}The helper class should mark Tuple2 as moveable (using the template metaprograming from my last post), while the Tuple2 itself would stay POD. But for some reason I couldn't get it to work so far Any hints?

Honza