Subject: FEATURE: T* ArrayIndex::Detach(int i) AND mini fix
Posted by kohait00 on Thu, 22 Apr 2010 09:06:14 GMT
View Forum Message <> Reply to Message

the Detach function available in Array:: and ArrayMap (recently added) should be added to ArrayIndex as well, right?

Index.h:229 should be added sth like

```
T      *PopDetach()                  { B::hash.Drop(1); return B::key.PopDetach(); }
T      *Detach(int i)               { B::hash.Remove(i); return B::key.Detach(i); }
```

and a mini fix: (maybe forgotten? standard param)

Index.h:52 should be

```
 void  Drop(int n = 1);
```

Subject: Re: FEATURE: T* ArrayIndex::Detach(int i) AND mini fix
Posted by mirek on Mon, 26 Apr 2010 08:18:30 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 22 April 2010 05:06the Detach function available in Array:: and ArrayMap (recently added) should be added to ArrayIndex as well, right?

Index.h:229 should be added sth like

```
T      *PopDetach()                  { B::hash.Drop(1); return B::key.PopDetach(); }
T      *Detach(int i)               { B::hash.Remove(i); return B::key.Detach(i); }
```

Well, why not, applied.

Frankly, IndexArray is provided just for completeness. I do not remember using it ever... It so happens that all potential keys tend to be moveable... (in fact, 98% of keys is either int or String anyway).

Quote:
and a mini fix: (maybe forgotten? standard param)

Index.h:52 should be

```
void  Drop(int n = 1);
```

Well, HashBase is just implementation issue that is not supposed to be used on its own... So its interface is irrelevant from client's perspective.

Mirek

Subject: Re: FEATURE: T* ArrayIndex::Detach(int i) AND mini fix
Posted by kohait00 on Mon, 26 Apr 2010 15:50:03 GMT
View Forum Message <> Reply to Message

same with me, i havent used ArrayIndex so far. in a recent case i was in need to have kind of a "dictionary" of data objects, which have their own GetHashValue, so i wanted to see if ArrayIndex is my coice to avoid ArrayMap usage like to save some writing.

```
OBJECT * obj;
...
amap.Add(obj->GetHashValue(), obj);
```

rather like this

```
OBJECT * obj;
...
ind.Add(obj);
```

but realized i have no means to provide a hash on its own, so i stayed with ArrayMap. and by the way had a bit of investigation of API. frankly, i really like the containers

---