Subject: SQL*, SQL&, SQL*:: ???
Posted by jeremyc on Sat, 01 Sep 2007 19:34:32 GMT
View Forum Message <> Reply to Message

I do not understand all of these that I see in code. Can someone tell me the difference between:

SQL * Select(...);
SQL & Select(...);
SQL * ::Select(...);

I've seen it in example code all three ways. Maybe more.

Thanks,

Jeremy

Subject: Re: SQL*, SQL&, SQL*:: ???
Posted by mirek on Sun, 02 Sep 2007 22:14:32 GMT
View Forum Message <> Reply to Message

In 99% of cases, you use the first one.

"SQL &" throws an exception on error. Usually, with transactional RDBMS, using transaction block over exceptions is easier (even if not completely optimal), so it less often used.

"SQL * ::Select" is a bug, should be "SQL * UPP::Select", in what example you have spotted it? This has to be used in case when there is method named "Select" (name clash problem resolved by fully qualifying the name).

Mirek

Subject: Re: SQL*, SQL&, SQL*:: ???
Posted by sergeynikitin on Thu, 27 Nov 2008 18:21:46 GMT
View Forum Message <> Reply to Message

I have seen similar use in HomeBudget.
But there it was ::Update.

Incidentally, all want more examples on databases and SQL.

Subject: Re: SQL*, SQL&, SQL*:: ???
Posted by unodgs on Thu, 27 Nov 2008 19:19:38 GMT
View Forum Message <> Reply to Message

sergeynikitin wrote on Thu, 27 November 2008 13:21I have seen similar use in HomeBudget.
But there it was ::Update.

Incidentally, all want more examples on databases and SQL.
:: Is not necessary, it is even wrong as all upp functions belogns to Upp namespace.
I often had in my apps Update method in which I had SQL * ::Update. Without scope selector I would have names conflict. I still use :: instead of UPP:: because it's shorter.

Subject: Re: SQL*, SQL&, SQL*:: ???
Posted by unodgs on Thu, 27 Nov 2008 19:24:01 GMT
View Forum Message <> Reply to Message

operator * calls Execute(), & calls ExecuteX which throws SqlExc if something is wrong. Useful if you do some operation in a transaction.