**ASP.NET性能优化之分布式Session**

作者：[Infoq博客](http://www.it165.net/pro/html/201111/1087.html)  发布日期：2011-11-03 11:10:11

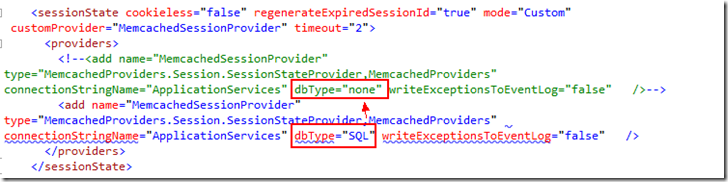
[**我来说两句(0)**](http://www.it165.net/pro/html/201111/1087.html#comment_iframe)[http://www.it165.net/statics/css/4.0/images/tousu.jpg](http://www.it165.net/index.php?m=formguide&c=index&a=show&formid=27)

**0**

**Tag标签：**[ASP.NET](http://s.it165.net/?t=13&q=ASP.NET" \t "_blank)  [性能优化](http://s.it165.net/?t=13&q=%D0%D4%C4%DC%D3%C5%BB%AF)  [Session](http://s.it165.net/?t=13&q=Session)

* 一：Memcached Providers

我使用的Memcached客户端是Memcached Providers，下载完毕后，你会发现Memcached Providers已经提供了对分布式Session的支持功能。如果你还不会使用Memcached Providers。Memcached Providers提供的示例是直接将SESSION存储在数据库，我们可以通过配置来将SESSION支持存储在分布式SESSION的内存中，即，将下文中的dbType由SQL修改为none。：



使用Memcached Providers提供的分布式Session没有任何特别之处，因为Memcached Providers提供的SessionStateProvider类型实现的是[ASP](http://www.it165.net/pro/webasp/" \t "_blank).NET中的 SessionStateStoreProviderBase这个抽象类，我们可以看到配置文件中指定了Session的处理类是 SessionStateProvider，所以，[ASP](http://www.it165.net/pro/webasp/" \t "_blank).NET在接受到客户端的请求后，会自觉滴使用SessionStateProvider来处理所有的SESSION，也正是这个类，完成了将SESSION读取和存储在Memcached中（如果设置了SQL，则会同步存储到SQLSERVER数据库）。

SESSION的设置和读取与传统没有任何区别，

Session["sname2"] = "sluminjxxi";

Session.Timeout = 2;

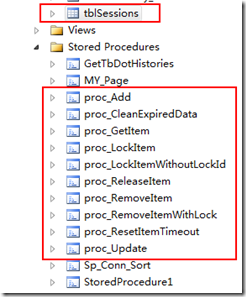
取：

Response.Write(Session["sname2"]);

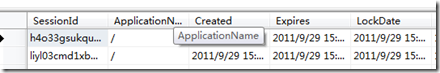
二：为什么要配置SQL

传统的SESSION的缺点，在仅使用dbType为none配置的时候都会存在。如Memcached的内存到达上限的时候会怎么办？Memcached使用LRU淘汰算法（最久未使用），在这里我们不需要去细究这个算法在Memcached内部到底是什么样一个机制，我们只需要知道，在内存紧张的时候，即使SESSION时间未到，Memcached也有可能把它干掉。所以，保险的做法是，在Memcached之下，再加上 SQLSERVER的持久化保存。如果缓存命中的，直接取缓存，如果缓存没命中的，则再到数据库中确认一次。当然，这样会带来一些性能损耗，但是却是更安全的做法。

Memcached Providers提供的下载文件中，提供了初始化SESSION的一些脚本，正确执行后，它会生成如下一个表tblSessions，及若干存储过程：

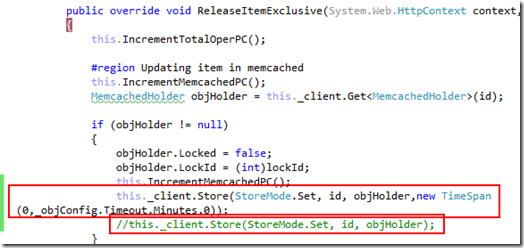


tblSessions保存的是就是单独的Session，如下：



四：Memcached Providers的一个BUG

在当前的Memcached Providers（1.2版本）中关于SessionStateProvider（29520-TRUNK）是有一个BUG（我已提交到 codeplex，相信他们的下一个版本应该能得到修正）的。如果我们测试SESSION失效时间，发现只要经过一次刷新后，就永远是20分钟（即默认）。这源于在ReleaseItemExclusive这个重载方法中（该方法用于释放对会话数据存储区中项的锁定），对于Session的重新存储没有加上过期时间，如下：



注释掉的是Memcached Providers提供的源码，而正确的应该是我修正过的上一条。使用修正过的DLL，一切圆满了。

五：采用数据库存储SESSION的可扩展问题

随着访问量的进一步上升（当然，到了这种程度，说明网站做的很很成功，绝大部分的网站是不需要考虑这一步的），即便我们使用了Memcached作缓存，使用单一的SQLSERVER存储SESSION仍旧带来了性能问题，在这种情况下，我们对于数据库的设计可以采用水平分区的架构，即根据某种算法（可以根据SESSIONID，或者用户名等）将SESSION存储到不同的数据库中。这个时候，如果我们仍旧使用Memcached Providers，那么必须进一步修改源码了，由原先支持单一SQLSERVER服务器，编程支持多个服务器。当然，如果不喜欢SQLSERVER，还可以修改为支持mysql、mongodb、任何自定义的KEY-VALUE框架等等，此为后话，暂且不表。