链滴

一个通用的计数限制解决方案

作者：loogn

原文链接：https://ld246.com/article/1561947640492

来源网站：链滴

场景：

1，一个用户在一天内最多修改3次密码；

2，一个手机号在五分钟内只能接受2次验证码；

3，一个用户在一个月内只能置顶一篇文章；

。。。。。。

像这样的场景应该是很常见的，每个系统中都可能用到，一个系统中可能多个地方都会用到，所以有要设计一个通用的方案来处理这种问题。

分析其本质，可以用一句话来概括：**一个事件在一个时间段内最多可以执行特定次数。**

模型设计：

```
/// <summary>
    /// 计数限制 /// </summary>
    public class CountLimit
    {
        [OrmLiteField(IsPrimaryKey = true, InsertIgnore = true)] public long Id { get; set; } /// <s
mmary>
        /// 唯一标识 /// </summary>
        public string UniqueId { get; set; } /// <summary>
        /// 用途 /// </summary>
        public byte UseType { get; set; } /// <summary>
        /// 已有计数 /// </summary>
        public int AlreadyCount { get; set; } /// <summary>
        /// 开始计数时间 /// </summary>
        public DateTime BeginTime { get; set; }
    }
```

模型中使用UniqueId和UseType来确定一个事件，比如手机号18500000000接受验证码，18500000000是UniqueId,接受验证码是UseType=1；

BeginTime是开始计时时间，如果是每个自然天的话，可能就是2018-5-11 00:00:00，注意，这么没结束时间；

AlreadyCount是从计时开始，已经发生的计数；

持久层方法：

这里的持久化使用sqlserver数据库，也可以使用类似redis的缓存数据库，如果那样的话，可以稍微一下模型，把uniqueId和useType合并成一个键，来代替id，这个就不多说了，用者自知。

```
public class D_CountLimit
   { public static CountLimit Single(byte useType, string uniqueId)
      { using (var db = DB.Open())
         { return db.SingleWhere<CountLimit>(DictBuilder.Assign("UniqueId", uniqueId).Assign
"UseType", useType));
         }
      } public static long Add(CountLimit m)
      { using (var db = DB.Open())
         { return db.Insert(m);
         }
      } public static int IncCount(long id)
      { using (var db = DB.Open())
         { return db.UpdateFieldById<CountLimit>("$AlreadyCount", "AlreadyCount+1", id);
         }
      } public static int ResetOne(long id, DateTime beginTime)
      { using (var db = DB.Open())
         { return db.UpdateById<CountLimit>(DictBuilder.Assign("AlreadyCount", 1).Assign("B
ginTime", beginTime), id);
         }
      }
   }
```

通用控制逻辑：

CheckLimit方法最后的三个参数，就是通用性的来源，上面说模型中没有结束时间，是因为控制权在后两个参数上。

比如时间段是自然天，那么duration=TimeSpan.FromDays(1)，beginTime=DateTime.Now.Date 如果从触发时起24小时之后为一天，beginTime=DateTime.Now即可，其他周期类推）；

```
   public class C_CountLimit
   { /// <summary>
      /// 检测计数公共方法 /// </summary>
      /// <param name="useType">使用类型</param>
      /// <param name="uniqueId">特定使用类型中唯一标识</param>
      /// <param name="limit">限制次数</param>
      /// <param name="duration">时间间隔</param>
      /// <param name="beginTime">当前间隔开始时间</param>
      /// <returns></returns>
      public static bool CheckLimit(byte useType, string uniqueId, int limit, TimeSpan duration,
DateTime? beginTime = null)
      { if (beginTime == null)
         {
            beginTime = DateTime.Now;
         } var m = D_CountLimit.Single(useType, uniqueId); //还没计数记录
         if (m == null)
         {
            D_CountLimit.Add(new CountLimit
```

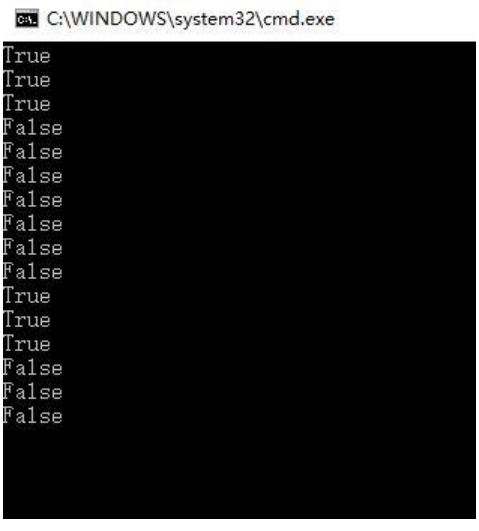
```
                {
                    AlreadyCount = 1,
                    BeginTime = beginTime.Value,
                    UniqueId = uniqueId,
                    UseType = useType
                });
            } else { //时间范围之内
                if (m.BeginTime <= DateTime.Now && m.BeginTime.Add(duration) > DateTime.No
)
                { //计数已满
                    if (m.AlreadyCount >= limit)
                    { return false;
                    } else { //增加计数
 D_CountLimit.IncCount(m.Id);
                    }
                } else { //不在时间范围之内，设置为一次计数
 D_CountLimit.ResetOne(m.Id, beginTime.Value);
                }
            } return true;
        }
    }
```

测试：

下面的例子是检测10秒内18500000000这个手机号只能发3次验证码（假设useType=1）

```
    while (true)
    { var result = C_CountLimit.CheckLimit(1, "18500000000", 3, TimeSpan.FromSeconds(10), Da
eTime.Now);
        Console.WriteLine(result);
        Thread.Sleep(1000);
    }
```

由于我们一秒执行一次，下图和我们的预期正好相符：



有些情况并没有什么时间周期，而是在整个系统之内只能执行有限的次数，这时只需把duration设置大一些就行了，比如设置为TimeSpan.FromDays(365000)，千年轮回，在1000年之后这个系统存不在便是个问题了！