**asp.net mvc 用Redis实现分布式集群共享Session**

|  |  |  |
| --- | --- | --- |
| [日期：2015-07-18] | 来源：Linux社区  作者：valen | [字体：[大](javascript:ContentSize(16)) [中](javascript:ContentSize(0)) [小](javascript:ContentSize(12))] |

1、这两天研究Redis搞分布式session问题，网上找的资料都是用ServiceStack.Redis来实现的，但是在做性能测试的时候发现最新的v4版本有限制每小时候最多请求6000次，因为官网开始商业化要收费了，好坑爹的说，还好我前期弄了个性能测试列子，不然上线以后出问题那就麻烦了。后面找了个NServiceKit.Redis（好像就是ServiceStack.Redis的v3版本）来替代v4的收费版。

2、解决方案是 Redis+cookie方式实现记录用户登录状态

cookie：存放用户的ID，这个ID是经过加密的，并且后台可以通过密钥解密。

Redis：key/value 方式存储，key存放比如：user\_1。  value存放用户实体对象。

3、先安装一个Redis，windows的版本在本地进行测试，后期上线更换linux系统的Redis替换一下ip就可以了。

4、添加一个Session管理类

public class SessionHelper  
    {  
        private const int secondsTimeOut = 60 \* 20;  //默认过期时间20分钟  单位秒

        public RedisHelper Redis = new RedisHelper(false);  
        public LoginUserInfo this[string key]  
        {  
            get  
            {  
                string webCookie = WebHelper.GetCookie(key);  
                if (webCookie == "")  
                {  
                    return null;  
                }  
                key = key + "\_" + SecureHelper.AESDecrypt(webCookie);

                //距离过期时间还有多少秒  
                long l = Redis.TTL(key);  
                if (l >= 0)  
                {  
                    Redis.Expire(key, secondsTimeOut);  
                }  
                  
                return Redis.Get<LoginUserInfo>(key);  
            }  
            set  
            {  
                SetSession(key, value);  
            }  
        }  
        public void SetSession(string key, LoginUserInfo value)  
        {  
            if (string.IsNullOrWhiteSpace(key))  
            {  
                throw new Exception("Key is Null or Epmty");  
            }  
            WebHelper.SetCookie(key, SecureHelper.AESEncrypt(value.ID.ToString()));  
            key = key + "\_" + value.ID;  
            Redis.Set<LoginUserInfo>(key, value, secondsTimeOut);  
        }

        /// <summary>  
        /// 移除Session  
        /// </summary>  
        /// <param name="key"></param>  
        /// <returns></returns>  
        public bool Remove(string key)  
        {  
            var rs = Redis.Remove(key + "\_" + SecureHelper.AESDecrypt(WebHelper.GetCookie(key)));  
            WebHelper.DeleteCookie(key);  
            return rs;  
        }  
              
    }

5、Redis操作类

public class RedisHelper : IDisposable  
    {  
      private RedisClient Redis = new RedisClient("127.0.0.1", 6379);  
        //缓存池  
        PooledRedisClientManager prcm = new PooledRedisClientManager();

        //默认缓存过期时间单位秒  
        public int secondsTimeOut = 20 \* 60;

        /// <summary>  
        /// 缓冲池  
        /// </summary>  
        /// <param name="readWriteHosts"></param>  
        /// <param name="readOnlyHosts"></param>  
        /// <returns></returns>  
        public static PooledRedisClientManager CreateManager(string[] readWriteHosts, string[] readOnlyHosts)  
        {  
            return new PooledRedisClientManager(readWriteHosts, readOnlyHosts,  
                new RedisClientManagerConfig  
                {  
                    MaxWritePoolSize = readWriteHosts.Length \* 5,  
                    MaxReadPoolSize = readOnlyHosts.Length \* 5,  
                    AutoStart = true,  
                });   
        }  
        /// <summary>  
        /// 构造函数  
        /// </summary>  
        /// <param name="OpenPooledRedis">是否开启缓冲池</param>  
        public RedisHelper(bool OpenPooledRedis = false)  
        {

            if (OpenPooledRedis)  
            {  
                prcm = CreateManager(new string[] { "127.0.0.1:6379" }, new string[] { "127.0.0.1:6379" });  
                Redis = prcm.GetClient() as RedisClient;  
            }  
        }  
        /// <summary>  
        /// 距离过期时间还有多少秒  
        /// </summary>  
        /// <param name="key"></param>  
        /// <returns></returns>  
        public long TTL(string key)  
        {  
            return Redis.Ttl(key);  
        }  
        /// <summary>  
        /// 设置过期时间  
        /// </summary>  
        /// <param name="key"></param>  
        /// <param name="timeout"></param>  
        public void Expire(string key,int timeout = 0)  
        {  
            if (timeout >= 0)  
            {  
                if (timeout > 0)  
                {  
                    secondsTimeOut = timeout;  
                }  
                Redis.Expire(key, secondsTimeOut);  
            }  
        }

        #region Key/Value存储  
        /// <summary>  
        /// 设置缓存  
        /// </summary>  
        /// <typeparam name="T"></typeparam>  
        /// <param name="key">缓存建</param>  
        /// <param name="t">缓存值</param>  
        /// <param name="timeout">过期时间，单位秒,-1：不过期，0：默认过期时间</param>  
        /// <returns></returns>  
        public bool Set<T>(string key, T t, int timeout = 0)  
        {  
            Redis.Set<T>(key, t);  
            if (timeout >= 0)  
            {  
                if (timeout > 0)  
                {  
                    secondsTimeOut = timeout;  
                }  
                Redis.Expire(key, secondsTimeOut);  
            }  
            return true;  
              
        }  
        /// <summary>  
        /// 获取  
        /// </summary>  
        /// <typeparam name="T"></typeparam>  
        /// <param name="key"></param>  
        /// <returns></returns>  
        public T Get<T>(string key)  
        {  
            return Redis.Get<T>(key);  
        }  
        /// <summary>  
        /// 删除  
        /// </summary>  
        /// <param name="key"></param>  
        /// <returns></returns>  
        public bool Remove(string key)  
        {  
            return Redis.Remove(key);  
        }  
        #endregion

        //释放资源  
        public void Dispose()  
        {  
            if (Redis != null)  
            {  
                Redis.Dispose();  
                Redis = null;  
            }  
            GC.Collect();

        }  
    }