一、引入ServiceStack

　　通过NuGET搜索ServiceStack，安装之后会有4个dll，如下图

二、启动Redis服务

这里按照上一篇博客主从复制的结果搭建Redis服务器。6379的是主服务器，6380的是从服务器。图我就不截了，上篇博客中已经有了。

1. 封装帮助类

1.配置文件

using System;

using System.Collections.Generic;

using System.Linq;using System.Text;

using System.Threading.Tasks;

using System.Configuration;

namespace RedisHelper

{

public sealed class RedisConfig : ConfigurationSection

{

public static string WriteServerConStr{

get {

return string.Format("{0},{1}","127.0.0.1:6379","127.0.0.1:6380");

}

}

public static string ReadServerConStr

{

get

{

return string.Format("{0}", "127.0.0.1:6379");

}

}

public static int MaxWritePoolSize

{

get

{

return 50;

}

}

public static int MaxReadPoolSize

{

get

{

return 200;

}

}

public static bool AutoStart

{

get

{

return true;

}

}

}

}

2.RedisManager管理类 主要管理维护服务端访问类

using ServiceStack.Redis;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RedisHelper

{

public class RedisManager

{

private static PooledRedisClientManager prcm;

/// <summary>

/// 静态构造方法，初始化链接池管理对象

/// </summary>

static RedisManager()

{

CreateManager();

}

/// <summary>

/// 创建链接池管理对象

/// </summary>

private static void CreateManager()

{

string[] WriteServerConStr = SplitString(RedisConfig.WriteServerConStr, ",");

string[] ReadServerConStr = SplitString(RedisConfig.ReadServerConStr, ",");

prcm = new PooledRedisClientManager(ReadServerConStr, WriteServerConStr,

new RedisClientManagerConfig

{

MaxWritePoolSize = RedisConfig.MaxWritePoolSize,

MaxReadPoolSize = RedisConfig.MaxReadPoolSize,

AutoStart = RedisConfig.AutoStart,

});

}

private static string[] SplitString(string strSource, string split)

{

return strSource.Split(split.ToArray());

}

/// <summary>

/// 客户端缓存操作对象

/// </summary>

public static IRedisClient GetClient()

{

if (prcm == null)

CreateManager();

return prcm.GetClient();

}

}

}

3.RedisBase类

using ServiceStack.Redis;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RedisHelper

{

/// <summary>

/// RedisBase类，是redis操作的基类，继承自IDisposable接口，主要用于释放内存

/// </summary>

public abstract class RedisBase : IDisposable

{

public static IRedisClient Core { get; private set; }

private bool \_disposed = false;

static RedisBase()

{

Core = RedisManager.GetClient();

}

protected virtual void Dispose(bool disposing)

{

if (!this.\_disposed)

{

if (disposing)

{

Core.Dispose();

Core = null;

}

}

this.\_disposed = true;

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

/// <summary>

/// 保存数据DB文件到硬盘

/// </summary>

public void Save()

{

Core.Save();

}

/// <summary>

/// 异步保存数据DB文件到硬盘

/// </summary>

public void SaveAsync()

{

Core.SaveAsync();

}

}

}

4.字符串、List、Hash等操作类

(1)string操作类

namespace RedisHelper

{

public class RedisString : RedisBase

{

#region 赋值

/// <summary>

/// 设置key的value

/// </summary>

public static bool Set(string key, string value)

{

return RedisBase.Core.Set<string>(key, value);

}

/// <summary>

/// 设置key的value并设置过期时间

/// </summary>

public static bool Set(string key, string value, DateTime dt)

{

return RedisBase.Core.Set<string>(key, value, dt);

}

/// <summary>

/// 设置key的value并设置过期时间

/// </summary>

public static bool Set(string key, string value, TimeSpan sp)

{

return RedisBase.Core.Set<string>(key, value, sp);

}

/// <summary>

/// 设置多个key/value

/// </summary>

public static void Set(Dictionary<string, string> dic)

{

RedisBase.Core.SetAll(dic);

}

#endregion

#region 追加

/// <summary>

/// 在原有key的value值之后追加value

/// </summary>

public static long Append(string key, string value)

{

return RedisBase.Core.AppendToValue(key, value);

}

#endregion

#region 获取值

/// <summary>

/// 获取key的value值

/// </summary>

public static string Get(string key)

{

return RedisBase.Core.GetValue(key);

}

/// <summary>

/// 获取多个key的value值

/// </summary>

public static List<string> Get(List<string> keys)

{

return RedisBase.Core.GetValues(keys);

}

/// <summary>

/// 获取多个key的value值

/// </summary>

public static List<T> Get<T>(List<string> keys)

{

return RedisBase.Core.GetValues<T>(keys);

}

#endregion

#region 获取旧值赋上新值

/// <summary>

/// 获取旧值赋上新值

/// </summary>

public string GetAndSetValue(string key, string value)

{

return RedisBase.Core.GetAndSetValue(key, value);

}

#endregion

#region 辅助方法

/// <summary>

/// 获取值的长度

/// </summary>

public static long GetCount(string key)

{

return RedisBase.Core.GetStringCount(key);

}

/// <summary>

/// 自增1，返回自增后的值

/// </summary>

public static long Incr(string key)

{

return RedisBase.Core.IncrementValue(key);

}

/// <summary>

/// 自增count，返回自增后的值

/// </summary>

public static double IncrBy(string key, double count)

{

return RedisBase.Core.IncrementValueBy(key, count);

}

/// <summary>

/// 自减1，返回自减后的值

/// </summary>

public static long Decr(string key)

{

return RedisBase.Core.DecrementValue(key);

}

/// <summary>

/// 自减count ，返回自减后的值

/// </summary>

/// <param name="key"></param>

/// <param name="count"></param>

/// <returns></returns>

public static long DecrBy(string key, int count)

{

return RedisBase.Core.DecrementValueBy(key, count);

}

#endregion

}

}

(2)List操作类

namespace RedisHelper

{

public class RedisList : RedisBase

{

#region 赋值

/// <summary>

/// 从左侧向list中添加值

/// </summary>

public static void LPush(string key, string value)

{

RedisBase.Core.PushItemToList(key, value);

}

/// <summary>

/// 从左侧向list中添加值，并设置过期时间

/// </summary>

public static void LPush(string key, string value, DateTime dt)

{

RedisBase.Core.PushItemToList(key, value);

RedisBase.Core.ExpireEntryAt(key, dt);

}

/// <summary>

/// 从左侧向list中添加值，设置过期时间

/// </summary>

public static void LPush(string key, string value, TimeSpan sp)

{

RedisBase.Core.PushItemToList(key, value);

RedisBase.Core.ExpireEntryIn(key, sp);

}

/// <summary>

/// 从左侧向list中添加值

/// </summary>

public static void RPush(string key, string value)

{

RedisBase.Core.PrependItemToList(key, value);

}

/// <summary>

/// 从右侧向list中添加值，并设置过期时间

/// </summary>

public static void RPush(string key, string value, DateTime dt)

{

RedisBase.Core.PrependItemToList(key, value);

RedisBase.Core.ExpireEntryAt(key, dt);

}

/// <summary>

/// 从右侧向list中添加值，并设置过期时间

/// </summary>

public static void RPush(string key, string value, TimeSpan sp)

{

RedisBase.Core.PrependItemToList(key, value);

RedisBase.Core.ExpireEntryIn(key, sp);

}

/// <summary>

/// 添加key/value

/// </summary>

public static void Add(string key, string value)

{

RedisBase.Core.AddItemToList(key, value);

}

/// <summary>

/// 添加key/value ,并设置过期时间

/// </summary>

public static void Add(string key, string value, DateTime dt)

{

RedisBase.Core.AddItemToList(key, value);

RedisBase.Core.ExpireEntryAt(key, dt);

}

/// <summary>

/// 添加key/value。并添加过期时间

/// </summary>

public static void Add(string key, string value, TimeSpan sp)

{

RedisBase.Core.AddItemToList(key, value);

RedisBase.Core.ExpireEntryIn(key, sp);

}

/// <summary>

/// 为key添加多个值

/// </summary>

public static void Add(string key, List<string> values)

{

RedisBase.Core.AddRangeToList(key, values);

}

/// <summary>

/// 为key添加多个值，并设置过期时间

/// </summary>

public static void Add(string key, List<string> values, DateTime dt)

{

RedisBase.Core.AddRangeToList(key, values);

RedisBase.Core.ExpireEntryAt(key, dt);

}

/// <summary>

/// 为key添加多个值，并设置过期时间

/// </summary>

public static void Add(string key, List<string> values, TimeSpan sp)

{

RedisBase.Core.AddRangeToList(key, values);

RedisBase.Core.ExpireEntryIn(key, sp);

}

#endregion

#region 获取值

/// <summary>

/// 获取list中key包含的数据数量

/// </summary>

public static long Count(string key)

{

return RedisBase.Core.GetListCount(key);

}

/// <summary>

/// 获取key包含的所有数据集合

/// </summary>

public static List<string> Get(string key)

{

return RedisBase.Core.GetAllItemsFromList(key);

}

/// <summary>

/// 获取key中下标为star到end的值集合

/// </summary>

public static List<string> Get(string key, int star, int end)

{

return RedisBase.Core.GetRangeFromList(key, star, end);

}

#endregion

#region 阻塞命令

/// <summary>

/// 阻塞命令：从list中keys的尾部移除一个值，并返回移除的值，阻塞时间为sp

/// </summary>

public static string BlockingPopItemFromList(string key, TimeSpan? sp)

{

return RedisBase.Core.BlockingDequeueItemFromList(key, sp);

}

/// <summary>

/// 阻塞命令：从list中keys的尾部移除一个值，并返回移除的值，阻塞时间为sp

/// </summary>

public static ItemRef BlockingPopItemFromLists(string[] keys, TimeSpan? sp)

{

return RedisBase.Core.BlockingPopItemFromLists(keys, sp);

}

/// <summary>

/// 阻塞命令：从list中keys的尾部移除一个值，并返回移除的值，阻塞时间为sp

/// </summary>

public static string BlockingDequeueItemFromList(string key, TimeSpan? sp)

{

return RedisBase.Core.BlockingDequeueItemFromList(key, sp);

}

/// <summary>

/// 阻塞命令：从list中keys的尾部移除一个值，并返回移除的值，阻塞时间为sp

/// </summary>

public static ItemRef BlockingDequeueItemFromLists(string[] keys, TimeSpan? sp)

{

return RedisBase.Core.BlockingDequeueItemFromLists(keys, sp);

}

/// <summary>

/// 阻塞命令：从list中key的头部移除一个值，并返回移除的值，阻塞时间为sp

/// </summary>

public static string BlockingRemoveStartFromList(string keys, TimeSpan? sp)

{

return RedisBase.Core.BlockingRemoveStartFromList(keys, sp);

}

/// <summary>

/// 阻塞命令：从list中key的头部移除一个值，并返回移除的值，阻塞时间为sp

/// </summary>

public static ItemRef BlockingRemoveStartFromLists(string[] keys, TimeSpan? sp)

{

return RedisBase.Core.BlockingRemoveStartFromLists(keys, sp);

}

/// <summary>

/// 阻塞命令：从list中一个fromkey的尾部移除一个值，添加到另外一个tokey的头部，并返回移除的值，阻塞时间为sp

/// </summary>

public static string BlockingPopAndPushItemBetweenLists(string fromkey, string tokey, TimeSpan? sp)

{

return RedisBase.Core.BlockingPopAndPushItemBetweenLists(fromkey, tokey, sp);

}

#endregion

#region 删除

/// <summary>

/// 从尾部移除数据，返回移除的数据

/// </summary>

public static string PopItemFromList(string key)

{

return RedisBase.Core.PopItemFromList(key);

}

/// <summary>

/// 移除list中，key/value,与参数相同的值，并返回移除的数量

/// </summary>

public static long RemoveItemFromList(string key, string value)

{

return RedisBase.Core.RemoveItemFromList(key, value);

}

/// <summary>

/// 从list的尾部移除一个数据，返回移除的数据

/// </summary>

public static string RemoveEndFromList(string key)

{

return RedisBase.Core.RemoveEndFromList(key);

}

/// <summary>

/// 从list的头部移除一个数据，返回移除的值

/// </summary>

public static string RemoveStartFromList(string key)

{

return RedisBase.Core.RemoveStartFromList(key);

}

#endregion

#region 其它

/// <summary>

/// 从一个list的尾部移除一个数据，添加到另外一个list的头部，并返回移动的值

/// </summary>

public static string PopAndPushItemBetweenLists(string fromKey, string toKey)

{

return RedisBase.Core.PopAndPushItemBetweenLists(fromKey, toKey);

}

#endregion

}

}

(3)Hash操作类

namespace RedisHelper

{

public class RedisHash : RedisBase

{

#region 添加

/// <summary>

/// 向hashid集合中添加key/value

/// </summary>

public static bool SetEntryInHash(string hashid, string key, string value)

{

return RedisBase.Core.SetEntryInHash(hashid, key, value);

}

/// <summary>

/// 如果hashid集合中存在key/value则不添加返回false，如果不存在在添加key/value,返回true

/// </summary>

public static bool SetEntryInHashIfNotExists(string hashid, string key, string value)

{

return RedisBase.Core.SetEntryInHashIfNotExists(hashid, key, value);

}

/// <summary>

/// 存储对象T t到hash集合中

/// </summary>

public static void StoreAsHash<T>(T t)

{

RedisBase.Core.StoreAsHash<T>(t);

}

#endregion

#region 获取

/// <summary>

/// 获取对象T中ID为id的数据。

/// </summary>

public static T GetFromHash<T>(object id)

{

return RedisBase.Core.GetFromHash<T>(id);

}

/// <summary>

/// 获取所有hashid数据集的key/value数据集合

/// </summary>

public static Dictionary<string, string> GetAllEntriesFromHash(string hashid)

{

return RedisBase.Core.GetAllEntriesFromHash(hashid);

}

/// <summary>

/// 获取hashid数据集中的数据总数

/// </summary>

public static long GetHashCount(string hashid)

{

return RedisBase.Core.GetHashCount(hashid);

}

/// <summary>

/// 获取hashid数据集中所有key的集合

/// </summary>

public static List<string> GetHashKeys(string hashid)

{

return RedisBase.Core.GetHashKeys(hashid);

}

/// <summary>

/// 获取hashid数据集中的所有value集合

/// </summary>

public static List<string> GetHashValues(string hashid)

{

return RedisBase.Core.GetHashValues(hashid);

}

/// <summary>

/// 获取hashid数据集中，key的value数据

/// </summary>

public static string GetValueFromHash(string hashid, string key)

{

return RedisBase.Core.GetValueFromHash(hashid, key);

}

/// <summary>

/// 获取hashid数据集中，多个keys的value集合

/// </summary>

public static List<string> GetValuesFromHash(string hashid, string[] keys)

{

return RedisBase.Core.GetValuesFromHash(hashid, keys);

}

#endregion

#region 删除

#endregion

/// <summary>

/// 删除hashid数据集中的key数据

/// </summary>

public static bool RemoveEntryFromHash(string hashid, string key)

{

return RedisBase.Core.RemoveEntryFromHash(hashid, key);

}

#region 其它

/// <summary>

/// 判断hashid数据集中是否存在key的数据

/// </summary>

public static bool HashContainsEntry(string hashid, string key)

{

return RedisBase.Core.HashContainsEntry(hashid, key);

}

/// <summary>

/// 给hashid数据集key的value加countby，返回相加后的数据

/// </summary>

public static double IncrementValueInHash(string hashid, string key, double countBy)

{

return RedisBase.Core.IncrementValueInHash(hashid, key, countBy);

}

#endregion

}

}

1. Set操作类

using System;using System.Collections.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;

namespace RedisHelper

{

public class RedisSet : RedisBase

{

#region 添加

/// <summary>

/// key集合中添加value值

/// </summary>

public static void Add(string key, string value)

{

RedisBase.Core.AddItemToSet(key, value);

}

/// <summary>

/// key集合中添加list集合

/// </summary>

public static void Add(string key, List<string> list)

{

RedisBase.Core.AddRangeToSet(key, list);

}

#endregion

#region 获取

/// <summary>

/// 随机获取key集合中的一个值

/// </summary>

public static string GetRandomItemFromSet(string key)

{

return RedisBase.Core.GetRandomItemFromSet(key);

}

/// <summary>

/// 获取key集合值的数量

/// </summary>

public static long GetCount(string key)

{

return RedisBase.Core.GetSetCount(key);

}

/// <summary>

/// 获取所有key集合的值

/// </summary>

public static HashSet<string> GetAllItemsFromSet(string key)

{

return RedisBase.Core.GetAllItemsFromSet(key);

}

#endregion

#region 删除

/// <summary>

/// 随机删除key集合中的一个值

/// </summary>

public static string PopItemFromSet(string key)

{

return RedisBase.Core.PopItemFromSet(key);

}

/// <summary>

/// 删除key集合中的value

/// </summary>

public static void RemoveItemFromSet(string key, string value)

{

RedisBase.Core.RemoveItemFromSet(key, value);

}

#endregion

#region 其它

/// <summary>

/// 从fromkey集合中移除值为value的值，并把value添加到tokey集合中

/// </summary>

public static void MoveBetweenSets(string fromkey, string tokey, string value)

{

RedisBase.Core.MoveBetweenSets(fromkey, tokey, value);

}

/// <summary>

/// 返回keys多个集合中的并集，返还hashset

/// </summary>

public static HashSet<string> GetUnionFromSets(string[] keys)

{

return RedisBase.Core.GetUnionFromSets(keys);

}

/// <summary>

/// keys多个集合中的并集，放入newkey集合中

/// </summary>

public static void StoreUnionFromSets(string newkey, string[] keys)

{

RedisBase.Core.StoreUnionFromSets(newkey, keys);

}

/// <summary>

/// 把fromkey集合中的数据与keys集合中的数据对比，fromkey集合中不存在keys集合中，则把这些不存在的数据放入newkey集合中

/// </summary>

public static void StoreDifferencesFromSet(string newkey, string fromkey, string[] keys)

{

RedisBase.Core.StoreDifferencesFromSet(newkey, fromkey, keys);

}

#endregion

}

}

(5)ZSet操作类

namespace RedisHelper

{

public class RedisZSet : RedisBase

{

#region 添加

/// <summary>

/// 添加key/value，默认分数是从1.多\*10的9次方以此递增的,自带自增效果

/// </summary>

public static bool AddItemToSortedSet(string key, string value)

{

return RedisBase.Core.AddItemToSortedSet(key, value);

}

/// <summary>

/// 添加key/value,并设置value的分数

/// </summary>

public static bool AddItemToSortedSet(string key, string value, double score)

{

return RedisBase.Core.AddItemToSortedSet(key, value, score);

}

/// <summary>

/// 为key添加values集合，values集合中每个value的分数设置为score

/// </summary>

public static bool AddRangeToSortedSet(string key, List<string> values, double score)

{

return RedisBase.Core.AddRangeToSortedSet(key, values, score);

}

/// <summary>

/// 为key添加values集合，values集合中每个value的分数设置为score

/// </summary>

public static bool AddRangeToSortedSet(string key, List<string> values, long score)

{

return RedisBase.Core.AddRangeToSortedSet(key, values, score);

}

#endregion

#region 获取

/// <summary>

/// 获取key的所有集合

/// </summary>

public static List<string> GetAllItemsFromSortedSet(string key)

{

return RedisBase.Core.GetAllItemsFromSortedSet(key);

}

/// <summary>

/// 获取key的所有集合，倒叙输出

/// </summary>

public static List<string> GetAllItemsFromSortedSetDesc(string key)

{

return RedisBase.Core.GetAllItemsFromSortedSetDesc(key);

}

/// <summary>

/// 获取可以的说有集合，带分数

/// </summary>

public static IDictionary<string, double> GetAllWithScoresFromSortedSet(string key)

{

return RedisBase.Core.GetAllWithScoresFromSortedSet(key);

}

/// <summary>

/// 获取key为value的下标值

/// </summary>

public static long GetItemIndexInSortedSet(string key, string value)

{

return RedisBase.Core.GetItemIndexInSortedSet(key, value);

}

/// <summary>

/// 倒叙排列获取key为value的下标值

/// </summary>

public static long GetItemIndexInSortedSetDesc(string key, string value)

{

return RedisBase.Core.GetItemIndexInSortedSetDesc(key, value);

}

/// <summary>

/// 获取key为value的分数

/// </summary>

public static double GetItemScoreInSortedSet(string key, string value)

{

return RedisBase.Core.GetItemScoreInSortedSet(key, value);

}

/// <summary>

/// 获取key所有集合的数据总数

/// </summary>

public static long GetSortedSetCount(string key)

{

return RedisBase.Core.GetSortedSetCount(key);

}

/// <summary>

/// key集合数据从分数为fromscore到分数为toscore的数据总数

/// </summary>

public static long GetSortedSetCount(string key, double fromScore, double toScore)

{

return RedisBase.Core.GetSortedSetCount(key, fromScore, toScore);

}

/// <summary>

/// 获取key集合从高分到低分排序数据，分数从fromscore到分数为toscore的数据

/// </summary>

public static List<string> GetRangeFromSortedSetByHighestScore(string key, double fromscore, double toscore)

{

return RedisBase.Core.GetRangeFromSortedSetByHighestScore(key, fromscore, toscore);

}

/// <summary>

/// 获取key集合从低分到高分排序数据，分数从fromscore到分数为toscore的数据

/// </summary>

public static List<string> GetRangeFromSortedSetByLowestScore(string key, double fromscore, double toscore)

{

return RedisBase.Core.GetRangeFromSortedSetByLowestScore(key, fromscore, toscore);

}

/// <summary>

/// 获取key集合从高分到低分排序数据，分数从fromscore到分数为toscore的数据，带分数

/// </summary>

public static IDictionary<string, double> GetRangeWithScoresFromSortedSetByHighestScore(string key, double fromscore, double toscore)

{

return RedisBase.Core.GetRangeWithScoresFromSortedSetByHighestScore(key, fromscore, toscore);

}

/// <summary>

/// 获取key集合从低分到高分排序数据，分数从fromscore到分数为toscore的数据，带分数

/// </summary>

public static IDictionary<string, double> GetRangeWithScoresFromSortedSetByLowestScore(string key, double fromscore, double toscore)

{

return RedisBase.Core.GetRangeWithScoresFromSortedSetByLowestScore(key, fromscore, toscore);

}

/// <summary>

/// 获取key集合数据，下标从fromRank到分数为toRank的数据

/// </summary>

public static List<string> GetRangeFromSortedSet(string key, int fromRank, int toRank)

{

return RedisBase.Core.GetRangeFromSortedSet(key, fromRank, toRank);

}

/// <summary>

/// 获取key集合倒叙排列数据，下标从fromRank到分数为toRank的数据

/// </summary>

public static List<string> GetRangeFromSortedSetDesc(string key, int fromRank, int toRank)

{

return RedisBase.Core.GetRangeFromSortedSetDesc(key, fromRank, toRank);

}

/// <summary>

/// 获取key集合数据，下标从fromRank到分数为toRank的数据，带分数

/// </summary>

public static IDictionary<string, double> GetRangeWithScoresFromSortedSet(string key, int fromRank, int toRank)

{

return RedisBase.Core.GetRangeWithScoresFromSortedSet(key, fromRank, toRank);

}

/// <summary>

/// 获取key集合倒叙排列数据，下标从fromRank到分数为toRank的数据，带分数

/// </summary>

public static IDictionary<string, double> GetRangeWithScoresFromSortedSetDesc(string key, int fromRank, int toRank)

{

return RedisBase.Core.GetRangeWithScoresFromSortedSetDesc(key, fromRank, toRank);

}

#endregion

#region 删除

/// <summary>

/// 删除key为value的数据

/// </summary>

public static bool RemoveItemFromSortedSet(string key, string value)

{

return RedisBase.Core.RemoveItemFromSortedSet(key, value);

}

/// <summary>

/// 删除下标从minRank到maxRank的key集合数据

/// </summary>

public static long RemoveRangeFromSortedSet(string key, int minRank, int maxRank)

{

return RedisBase.Core.RemoveRangeFromSortedSet(key, minRank, maxRank);

}

/// <summary>

/// 删除分数从fromscore到toscore的key集合数据

/// </summary>

public static long RemoveRangeFromSortedSetByScore(string key, double fromscore, double toscore)

{

return RedisBase.Core.RemoveRangeFromSortedSetByScore(key, fromscore, toscore);

}

/// <summary>

/// 删除key集合中分数最大的数据

/// </summary>

public static string PopItemWithHighestScoreFromSortedSet(string key)

{

return RedisBase.Core.PopItemWithHighestScoreFromSortedSet(key);

}

/// <summary>

/// 删除key集合中分数最小的数据

/// </summary>

public static string PopItemWithLowestScoreFromSortedSet(string key)

{

return RedisBase.Core.PopItemWithLowestScoreFromSortedSet(key);

}

#endregion

#region 其它

/// <summary>

/// 判断key集合中是否存在value数据

/// </summary>

public static bool SortedSetContainsItem(string key, string value)

{

return RedisBase.Core.SortedSetContainsItem(key, value);

}

/// <summary>

/// 为key集合值为value的数据，分数加scoreby，返回相加后的分数

/// </summary>

public static double IncrementItemInSortedSet(string key, string value, double scoreBy)

{

return RedisBase.Core.IncrementItemInSortedSet(key, value, scoreBy);

}

/// <summary>

/// 获取keys多个集合的交集，并把交集添加的newkey集合中，返回交集数据的总数

/// </summary>

public static long StoreIntersectFromSortedSets(string newkey, string[] keys)

{

return RedisBase.Core.StoreIntersectFromSortedSets(newkey, keys);

}

/// <summary>

/// 获取keys多个集合的并集，并把并集数据添加到newkey集合中，返回并集数据的总数

/// </summary>

public static long StoreUnionFromSortedSets(string newkey, string[] keys)

{

return RedisBase.Core.StoreUnionFromSortedSets(newkey, keys);

}

#endregion

}

}

四、操作类使用测试

class Program

{

static void Main(string[] args)

{

string key = "Users";

RedisBase.Core.FlushAll();

RedisBase.Core.AddItemToList(key, "cuiyanwei");

RedisBase.Core.AddItemToList(key, "xiaoming");

RedisBase.Core.Add<string>("mykey", "123456");

RedisString.Set("mykey1","abcdef");

Console.ReadLine();

}

}