C# Dictionary用法总结

1、用法1：　常规用

　　增加键值对之前需要判断是否存在该键，如果已经存在该键而且不判断，将抛出异常。所以这样每次都要进行判断，很麻烦，在备注里使用了一个扩展方法

|  |
| --- |
| public static void DicSample1()  {        Dictionary<String, String> pList = new Dictionary<String, String>();      try      {          if (pList.ContainsKey("Item1") == false)          {              pList.Add("Item1", "ZheJiang");          }          if (pList.ContainsKey("Item2")== false)          {              pList.Add("Item2", "ShangHai");          }          else          {              pList["Item2"] = "ShangHai";          }          if (pList.ContainsKey("Item3") == false)          {              pList.Add("Item3", "BeiJiang");          }        }      catch (System.Exception e)      {          Console.WriteLine("Error: {0}", e.Message);      }          //判断是否存在相应的key并显示      if (pList.ContainsKey("Item1"))      {          Console.WriteLine("Output: " + pList["Item1"]);      }        //遍历Key      foreach (var key in pList.Keys)      {          Console.WriteLine("Output Key: {0}", key);      }        //遍历Value      foreach (String value in pList.Values)      {          Console.WriteLine("Output Value: {0}", value);      }      //遍历Key和Value      foreach (var dic in pList)      {          Console.WriteLine("Output Key : {0}, Value : {1} ", dic.Key, dic.Value);      }  } |

　　2、用法2：Dictionary的Value为一个数组

|  |
| --- |
| /// <summary>  /// Dictionary的Value为一个数组  /// </summary>   public static void DicSample2()   {       Dictionary<String, String[]> dic = new Dictionary<String, String[]>();       String[] ZheJiang =  { "Huzhou", "HangZhou", "TaiZhou" };       String[] ShangHai = { "Budong", "Buxi" };       dic.Add("ZJ", ZheJiang);       dic.Add("SH", ShangHai);       Console.WriteLine("Output :" + dic["ZJ"][0]);   } |

　　3、用法3： Dictionary的Value为一个类

|  |
| --- |
| //Dictionary的Value为一个类  public static void DicSample3()   {       Dictionary<String, Student> stuList = new Dictionary<String, Student>();       Student stu = null;       for (int i = 0; i < 3; i++ )       {           stu = new Student();           stu.Name = i.ToString();           stu.Name = "StuName" + i.ToString();           stuList.Add(i.ToString(), stu);       }         foreach (var student in stuList)       {           Console.WriteLine("Output : Key {0}, Num : {1}, Name {2}", student.Key, student.Value.Name, student.Value.Name);       }   } |

Student类：

|  |
| --- |
| public class Student  {      public String Num { get; set; }      public String Name { get; set; }  } |

 4 备注：Dictionary的扩展方法使用

|  |
| --- |
| /// <summary>  /// Dictionary的扩展方法使用  /// </summary>   public static void DicSample4()   {       //1)普通调用       Dictionary<int, String> dict = new Dictionary<int, String>();       DictionaryExtensionMethodClass.TryAdd(dict, 1, "ZhangSan");       DictionaryExtensionMethodClass.TryAdd(dict, 2, "WangWu");       DictionaryExtensionMethodClass.AddOrPeplace(dict, 3, "WangWu");       DictionaryExtensionMethodClass.AddOrPeplace(dict, 3, "ZhangWu");       DictionaryExtensionMethodClass.TryAdd(dict, 2, "LiSi");         //2)TryAdd 和 AddOrReplace 这两个方法具有较强自我描述能力，用起来很省心，而且也简单：       dict.AddOrPeplace(20, "Orange");       dict.TryAdd(21, "Banana");       dict.TryAdd(22, "apple");         //3)像Linq或jQuery一样连起来写       dict.TryAdd(10, "Bob")           .TryAdd(11, "Tom")           .AddOrPeplace(12, "Jom");         //4) 获取值       String F = "Ba";       dict.TryGetValue(31, out F);       Console.WriteLine("F : {0}",F);         foreach (var dic in dict)       {           Console.WriteLine("Output : Key : {0}, Value : {1}", dic.Key, dic.Value);       }       //5)下面是使用GetValue获取值       var v1 = dict.GetValue(111,null);       var v2 = dict.GetValue(10,"abc");         //6)批量添加       var dict1 = new Dictionary<int,int>();       dict1.AddOrPeplace(3, 3);       dict1.AddOrPeplace(5, 5);         var dict2 = new Dictionary<int, int>();       dict2.AddOrPeplace(1, 1);       dict2.AddOrPeplace(4, 4);       dict2.AddRange(dict1, false);   } |

　　扩展方法所在的类

|  |
| --- |
| public static class DictionaryExtensionMethodClass  {      /// <summary>      /// 尝试将键和值添加到字典中：如果不存在，才添加；存在，不添加也不抛导常      /// </summary>      public static Dictionary<TKey, TValue> TryAdd<TKey, TValue>(this Dictionary<TKey, TValue> dict, TKey key, TValue value)      {          if (dict.ContainsKey(key) == false)              dict.Add(key, value);          return dict;      }        /// <summary>      /// 将键和值添加或替换到字典中：如果不存在，则添加；存在，则替换      /// </summary>      public static Dictionary<TKey, TValue> AddOrPeplace<TKey, TValue>(this Dictionary<TKey, TValue> dict, TKey key, TValue value)      {          dict[key] = value;          return dict;      }        /// <summary>      /// 获取与指定的键相关联的值，如果没有则返回输入的默认值      /// </summary>      public static TValue GetValue<TKey, TValue>(this Dictionary<TKey, TValue> dict, TKey key, TValue defaultValue)      {          return dict.ContainsKey(key)?dict[key] : defaultValue;      }        /// <summary>      /// 向字典中批量添加键值对      /// </summary>      /// <param name="replaceExisted">如果已存在，是否替换</param>      public static Dictionary<TKey, TValue> AddRange<TKey, TValue>(this Dictionary<TKey, TValue> dict, IEnumerable<KeyValuePair<TKey, TValue>> values, bool replaceExisted)      {          foreach (var item in values)          {              if (dict.ContainsKey(item.Key) == false || replaceExisted)                  dict[item.Key] = item.Value;          }          return dict;      }      } |

命名空间

System.Collections

名称

哈希表(Hashtable)

描述

用于处理和表现类似keyvalue的键值对，其中key通常可用来快速查找，同时key是区分大小写；value用于存储对应于key的值。Hashtable中keyvalue键值对均为object类型，所以Hashtable可以支持任何类型的keyvalue键值对.

二,哈希表的简单操作

Hashtable hshTable = new Hashtable(); //  创建哈希表  
hshTable .Add("Person1",  "zhanghf");  //  往哈希表里添加键值对  
hshTable .Clear();  //移除哈希表里所有的键值对  
hshTable .Contains("Person1");   //判断哈希表里是否包含该键  
string name = (string)hshTable["Person1"].ToString(); //取哈希表里指定键的值  
hshTable.Remove("Person1"); //  删除哈希表里指定键的键值对  
IDictionaryEnumerator en = hshTable.GetEnumerator();  //  遍历哈希表所有的键,读出相应的值  
while (en.MoveNext())  
            {  
               string str = en.Value.ToString();   
            }

下面控制台程序将包含以上所有操作：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | class Program      {          static void Main(string[] args)          {              // 创建一个Hashtable实例              Hashtable ht=new Hashtable();                // 添加keyvalue键值对              ht.Add("A","1");              ht.Add("B","2");              ht.Add("C","3");              ht.Add("D","4");                // 遍历哈希表              foreach (DictionaryEntry de in ht)              {                  Console.WriteLine("Key -- {0}; Value --{1}.", de.Key, de.Value);              }                // 哈希表排序              ArrayList akeys=new ArrayList(ht.Keys);              akeys.Sort();              foreach (string skey in akeys)              {                  Console.WriteLine("{0, -15} {1, -15}", skey, ht[skey]);                }                // 判断哈希表是否包含特定键,其返回值为true或false              if (ht.Contains("A"))                Console.WriteLine(ht["A"]);                // 给对应的键赋值              ht["A"] ="你好";                // 移除一个keyvalue键值对              ht.Remove("C");                // 遍历哈希表              foreach (DictionaryEntry de in ht)              {                  Console.WriteLine("Key -- {0}; Value --{1}.", de.Key, de.Value);              }                // 移除所有元素              ht.Clear();                // 此处将不会有任何输出              Console.WriteLine(ht["A"]);              Console.ReadKey();          }      } |