

По каким-то причинам Microsoft решила сделать класс Dictionary<TKey, TValue> не поддерживающим XML-сериализацию.

Т.е. код типа:

```
1 Dictionary<int, string> dict = new Dictionary<int, string>();
2 dict.Add(1, "aa");
3 dict.Add(2, "bb");
4 XmlSerializer xmlSerializer = new XmlSerializer(typeof(Dictionary<int, string>));
5 using (FileStream fs = new FileStream("test.xml", FileMode.CreateNew))
6 {
7     xmlSerializer.Serialize(fs, dict);
8 }
```

выполняться не будет. Будет отображена информация об ошибке вида:

«Тип System.Collections.Generic.Dictionary`2[...] не поддерживается, т.к. он реализует IDictionary.»

Такой расклад нас не устраивает...

Будем допиливать стандартный класс для поддержки сериализации.

1. Создаем класс наследник от Dictionary, реализующий интерфейс IXmlSerializable:

```
1 [XmlRoot("dictionary")]
2 public class SerializableDictionary<TKey, TValue> : Dictionary<TKey, TValue>,
3   IXmlSerializable
```

2. И, собственно, реализуем интерфейс:

2.1. Схема нам не нужна:

```
1 public XmlSchema GetSchema()
2 {
3     return null;
4 }
```

2.2. Чтение xml:

```

1 public void ReadXml(XmlReader reader)
2 {
3     var keySerializer = new XmlSerializer(typeof(TKey));
4     var valueSerializer = new XmlSerializer(typeof(TValue));
5     bool wasEmpty = reader.IsEmptyElement;
6     reader.Read();
7     if (wasEmpty) return;
8
9     while (reader.NodeType != XmlNodeType.EndElement)
10    {
11        reader.ReadStartElement("item");
12        reader.ReadStartElement("key");
13        var key = (TKey)keySerializer.Deserialize(reader);
14        reader.ReadEndElement();
15        reader.ReadStartElement("value");
16        var value = (TValue)valueSerializer.Deserialize(reader);
17        reader.ReadEndElement();
18        Add(key, value);
19        reader.ReadEndElement();
20        reader.MoveToContent();
21    }
22    reader.ReadEndElement();
23 }

```

2.3. Запись xml:

```

1 public void WriteXml(XmlWriter writer)
2 {
3     var keySerializer = new XmlSerializer(typeof(TKey));
4     var valueSerializer = new XmlSerializer(typeof(TValue));
5
6     foreach (TKey key in Keys)
7     {
8         writer.WriteStartElement("item");
9         writer.WriteStartElement("key");
10        keySerializer.Serialize(writer, key);
11        writer.WriteEndElement();
12        writer.WriteStartElement("value");
13        TValue value = this[key];
14        valueSerializer.Serialize(writer, value);
15        writer.WriteEndElement();
16        writer.WriteEndElement();
17    }
18 }

```

Все! Класс готов.

Проверяем:

```

1 Dictionary<int, string> dict = new SerializableDictionary<int, string>();
2 dict.Add(1, "aa");
3 dict.Add(2, "bb");
4 XmlSerializer xmlSerializer = new XmlSerializer(typeof(SerializableDictionary<int,
5 string>));
6 using (FileStream fs = new FileStream("test.xml", FileMode.CreateNew))
7 {
8     xmlSerializer.Serialize(fs, dict);
9 }

```

Работает! На выходе получится xml, такого вида:

```

1 <?xml version="1.0"?>
2 <dictionary>
3   <item>
4     <key>
5       <int>1</int>
6     </key>
7     <value>
8       <string>aa</string>
9     </value>
10  </item>
11  <item>
12    <key>
13      <int>2</int>
14    </key>
15    <value>
16      <string>bb</string>
17    </value>
18  </item>
19 </dictionary>

```

На всякий случай прикладываю код класса: [Скачать](#)

Примечание.

XmlSerializer.Serialize() может принимать не только FileStream, но и любой другой Stream, TextWriter, XmlWriter.

Для совместимости класса с LINQ, необходимо наличие метода ToDictionary.

ToDictionary — это обычный extension метод.

Можно его реализовать например так:

```

1 public static class ExtensionMethods
2 {
3     public static SerializableDictionary<TKey, TValue> ToSerializableDictionary<TKey, TValue, TSource>(this
4 IEnumerable<TSource> source, Func<TSource, TKey> keySelector, Func<TSource, TValue> elementSelector)
5     {
6         var d = new SerializableDictionary<TKey, TValue>();
7         foreach (TSource element in source)
8             d.Add(keySelector(element), elementSelector(element));
9         return d;
10    }
11 }

```

и использовать в LINQ:

```
1 var someList = new List<string> {"1", "2", "3", "4"};  
2 var sDict = someList  
3     .Where(x => x != null)  
4     .OrderByDescending(x => x)  
5     .ToSerializableDictionary(key => int.Parse(key), value => value + "qwe");
```

Информация взята со страницы dev-notes.ru