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

Т.е. код типа:

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

выполняться не будет. Будет отображаена 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:

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

2. И, собственно, реализуем интерфейс:
2.1. Схема нам не нужна:

```
public XmlSchema GetSchema()
{
    return null;
}
```

2.2. Чтение xml:

```
public void ReadXml(XmlReader reader)
{
    var keySerializer = new XmlSerializer(typeof(TKey));
    var valueSerializer = new XmlSerializer(typeof(TValue));
    bool wasEmpty = reader.IsEmptyElement;
    reader.Read();
    if (wasEmpty) return;

    while (reader.NodeType != XmlNodeType.EndElement)
    {
        reader.ReadStartElement("item");
        reader.ReadStartElement("key");
        var key = (TKey)keySerializer.Deserialize(reader);
        reader.ReadEndElement();
        reader.ReadStartElement("value");
        var value = (TValue)valueSerializer.Deserialize(reader);
        reader.ReadEndElement();
        Add(key, value);
        reader.ReadEndElement();
        reader.MoveToContent();
    }
    reader.ReadEndElement();
}
```

2.3. Запись xml:

```
public void WriteXml(XmlWriter writer)
{
    var keySerializer = new XmlSerializer(typeof(TKey));
    var valueSerializer = new XmlSerializer(typeof(TValue));

    foreach (TKey key in Keys)
    {
        writer.WriteStartElement("item");
        writer.WriteStartElement("key");
        keySerializer.Serialize(writer, key);
        writer.WriteEndElement();
        writer.WriteStartElement("value");
        TValue value = this[key];
        valueSerializer.Serialize(writer, value);
        writer.WriteEndElement();
        writer.WriteEndElement();
    }
}
```

Все! Класс готов.
Проверяем:

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

Работает! На выходе получится xml, такого вида:

```
<?xml version="1.0"?>
<dictionary>
  <item>
    <key>
      <int>1</int>
    </key>
    <value>
      <string>aa</string>
    </value>
  </item>
  <item>
    <key>
      <int>2</int>
    </key>
    <value>
      <string>bb</string>
    </value>
  </item>
</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 ToDictonary.
ToDictionary — это обычный extension метод.
Можно его реализовать например так:

```
public static class ExtensionMethods
{
    public static SerializableDictionary<TKey, TValue> ToSerializableDictionary<TKey, TValue, TSource>(this
IEnumerable<TSource> source, Func<TSource, TKey> keySelector, Func<TSource, TValue> elementSelector)
    {
        var d = new SerializableDictionary<TKey, TValue>();
        foreach (TSource element in source)
            d.Add(keySelector(element), elementSelector(element));
        return d;
    }
}
```

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