序列化：串行化，是.NET运行时环境用来支持用户自定义类型的流化机制。可以以某种存储形成自定义对象持久化，或者将这种对象从一个地方传输到另一个地方。

.NET提供的串行化方式：

1. 使用BinaryFormatter进行串行化

例：程序serialize\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// 序列化：BinaryFormatter

using System;

using System.Collections.Generic;

using System.Linq;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading.Tasks;

namespace serialize\_test1

{

[Serializable]

public class ClassToSerialize

{

private int id;

private string name;

// 不对Sex字段进行序列化

[NonSerialized]

private string sex;

public int ID

{

get { return id; }

}

public string Name

{

get { return name; }

}

public string Sex

{

get { return sex; }

}

public ClassToSerialize()

{

}

public ClassToSerialize(int id, string name, string Sex)

{

this.id = id;

this.name = name;

this.sex = Sex;

}

}

class SerializeTest

{

// 序列化

static void SerializeNow()

{

ClassToSerialize obj = new ClassToSerialize(100, "Name", "男");

FileStream stream = new FileStream("temp.dat", FileMode.Create);

BinaryFormatter formatter = new BinaryFormatter();

formatter.Serialize(stream, obj);

stream.Close();

}

// 反序列化

static void DeSerializeNow()

{

ClassToSerialize obj = new ClassToSerialize();

FileStream stream = new FileStream("temp.dat", FileMode.Open, FileAccess.Read);

BinaryFormatter formatter = new BinaryFormatter();

obj = formatter.Deserialize(stream) as ClassToSerialize;

stream.Close();

Console.WriteLine("DeSerialize obj Name " + obj.Name);

Console.WriteLine("DeSerialize obj ID " + obj.ID);

// 未进行序列化的字段反序列化得到的值为null

if (obj.Sex == null)

Console.WriteLine("DeSerialize obj Sex is null");

}

static void Main(string[] args)

{

Console.WriteLine("Start Serialize ");

SerializeNow();

Console.WriteLine("Start DeSerialize");

DeSerializeNow();

}

}

}

输出结果为：

Start Serialize

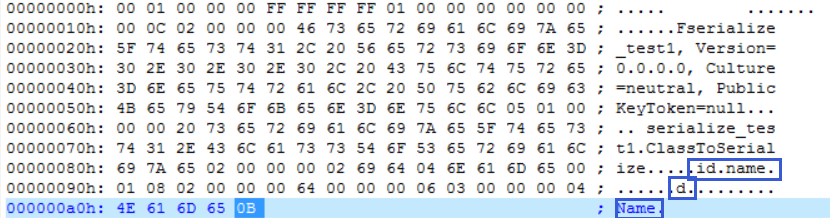
Start DeSerialize

DeSerialize obj Name Name

DeSerialize obj ID 100

DeSerialize obj Sex is null

其中序列化的结果为：



1. 使用XmlSerialize进行串行化

例：程序serialize\_test2

// Copyright 2016.刘珅珅

// author：刘珅珅

// 序列化：XmlSerializer

using System;

using System.Collections.Generic;

using System.Linq;

using System.IO;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Serialization;

namespace serialize\_test2

{

public class Person

{

private string name;

// 只读属性无法被Xml序列化

public string Name

{

get { return name; }

}

// 序列化非基本类型的数组时，必须

// 这样指明类型

[XmlElement(Type = typeof(Course))]

public Course[] course;

public Person() { }

public Person(string name)

{

this.name = name;

}

}

public class Course

{

private string name;

private string description;

public string Name

{

get { return name; }

set { name = value; }

}

public string Description

{

get { return description; }

set { description = value; }

}

public Course() {}

public Course(string name, string description)

{

this.name = name;

this.description = description;

}

}

class SerializeTest

{

// XML序列化

static void XMLSerialize()

{

Person p = new Person("cyj");

p.course = new Course[2];

p.course[0] = new Course("英语", "交流工具");

p.course[1] = new Course("数学", "自然科学");

XmlSerializer xs = new XmlSerializer(p.GetType());

Stream stream = new FileStream("temp.xml", FileMode.Create, FileAccess.Write);

xs.Serialize(stream, p);

stream.Close();

}

// 反序列化

static void XMLDeserialize()

{

XmlSerializer xs = new XmlSerializer(typeof(Person));

Stream stream = new FileStream("temp.xml", FileMode.Open, FileAccess.Read);

Person p = xs.Deserialize(stream) as Person;

stream.Close();

if (p.Name == null)

Console.WriteLine("XML Deserialize Name is null");

foreach (var c in p.course)

{

Console.WriteLine("XML Deserialize course name" + c.Name);

Console.WriteLine("XML Deserialize course description" + c.Description);

}

}

static void Main(string[] args)

{

XMLSerialize();

XMLDeserialize();

}

}

}

输出结果为：

XML Deserialize Name is null

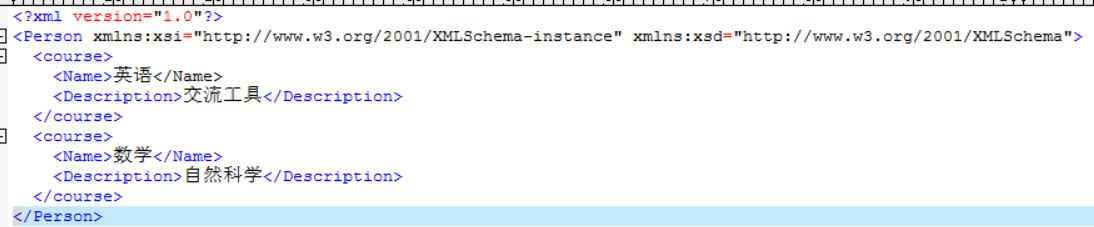
XML Deserialize course name英语

XML Deserialize course description交流工具

XML Deserialize course name数学

XML Deserialize course description自然科学

其中序列化的结果为：



使用XML序列化时要注意：

1. 要序列化的类必须有默认的无参构造函数
2. 方法不能被序列化
3. 索引器、私有字段或只读属性（只读集合属性除外）不能被序列化
4. 枚举变量可序列化为字符串
5. 非基本类型对象，必须使用类似于[XmlElement]这样的特征进行说明
6. 有些类无法进行XML序列化，可以考虑进行转换或二进制序列化