

只为成功找方法，不为失败找借口！

Java基础学习总结——Java对象的序列化和反序列化

一、序列化和反序列化的概念

把对象转换为字节序列的过程称为对象的序列化。  
把字节序列恢复为对象的过程称为对象的反序列化。

对象的序列化主要有两种用途：

- 1）把对象的字节序列永久地保存到硬盘上，通常存放在一个文件中；
- 2）在网络上传送对象的字节序列。

在很多应用中，需要对某些对象进行序列化，让它们离开内存空间，入住物理硬盘，以便长期保存。比如最常见的是Web服务器中的Session对象，当有 10万用户并发访问，就有可能出现10万个Session对象，内存可能吃不消，于是Web容器就会把一些seesion先序列化到硬盘中，等要用了，再把保存在硬盘中的对象还原到内存中。

当两个进程在进行远程通信时，彼此可以发送各种类型的数据。无论是何种类型的数据，都会以二进制序列的形式在网络上传送。发送方需要把这个Java对象转换为字节序列，才能在网络上传送；接收方则需要把字节序列再恢复为Java对象。

二、JDK类库中的序列化API

java.io.ObjectOutputStream代表对象输出流，它的writeObject(Object obj)方法可对参数指定的obj对象进行序列化，把得到的字节序列写到一个目标输出流中。

java.io.ObjectInputStream代表对象输入流，它的readObject()方法从一个源输入流中读取字节序列，再把它们反序列化为一个对象，并将其返回。

只有实现了Serializable和Externalizable接口的类的对象才能被序列化。Externalizable接口继承自 Serializable接口，实现Externalizable接口的类完全由自身来控制序列化的行为，而仅实现Serializable接口的类可以 采用默认的序列化方式。

对象序列化包括如下步骤：

- 1）创建一个对象输出流，它可以包装一个其他类型的目标输出流，如文件输出流；
- 2）通过对象输出流的writeObject()方法写对象。

对象反序列化的步骤如下：

- 1）创建一个对象输入流，它可以包装一个其他类型的源输入流，如文件输入流；
- 2）通过对象输入流的readObject()方法读取对象。

对象序列化和反序列范例：

定义一个Person类，实现Serializable接口



```
1 import java.io.Serializable;
2
```

导航

博客园 首页 联系 订阅 管理

≤ 2018年1月 ≥

日	一	二	三	四	五	六
31	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	1	2	3
4	5	6	7	8	9	10

公告

昵称：[孤傲苍狼](#)  
园龄：[6年9个月](#)  
粉丝：[12469](#)  
关注：[90](#)  
[+加关注](#)

统计

随笔 - 275 文章 - 0 评论 - 2884

搜索

找找看

谷歌搜索

常用链接

[我的随笔](#)  
[我的评论](#)  
[我的参与](#)  
[最新评论](#)  
[我的标签](#)

我的标签

[JavaWeb学习总结](#)(63)  
[java基础总结](#)(29)  
[Javascript学习总结](#)(27)  
[Snmp学习总结](#)(15)  
[MyEclipse使用总结](#)(13)  
[Android开发学习总结](#)(13)  
[Maven学习总结](#)(12)  
[MySQL学习总结](#)(9)  
[WebLogic使用总结](#)(9)  
[NginX学习总结](#)(8)  
[更多](#)

随笔分类

[AJAX](#)(2)  
[Android开发](#)(12)  
[ASP.NET](#)(2)  
[C#](#)(8)

```
3 /**
4  * <p>ClassName: Person<p>
5  * <p>Description:测试对象序列化和反序列化<p>
6  * @author xudp
7  * @version 1.0 V
8  * @createTime 2014-6-9 下午02:33:25
9  */
10 public class Person implements Serializable {
11
12     /**
13      * 序列化ID
14      */
15     private static final long serialVersionUID = -5809782578272943999L;
16     private int age;
17     private String name;
18     private String sex;
19
20     public int getAge() {
21         return age;
22     }
23
24     public String getName() {
25         return name;
26     }
27
28     public String getSex() {
29         return sex;
30     }
31
32     public void setAge(int age) {
33         this.age = age;
34     }
35
36     public void setName(String name) {
37         this.name = name;
38     }
39
40     public void setSex(String sex) {
41         this.sex = sex;
42     }
43 }
```



### 序列化和反序列化Person类对象



```
1 import java.io.File;
2 import java.io.FileInputStream;
3 import java.io.FileNotFoundException;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.ObjectInputStream;
7 import java.io.ObjectOutputStream;
8 import java.text.MessageFormat;
9
10 /**
```

[CSS学习总结](#)[C语言学习总结](#)[FusionCharts](#)[H2数据库学习使用总结\(3\)](#)[Hessian\(2\)](#)[Hibernate](#)[Highcharts](#)[HTML\(1\)](#)[html5学习使用总结](#)[Html学习总结](#)[Java\(13\)](#)[JavaScript\(27\)](#)[JavaWeb学习总结\(55\)](#)[Java基础加强总结\(3\)](#)[java基础面试题\(1\)](#)[java基础总结\(28\)](#)[Java监控](#)[Java事务处理](#)[java字节码的处理技术](#)[JFinal学习研究](#)[JNDI\(3\)](#)[Jquery EasyUI学习使用总结\(7\)](#)[jwebap](#)[kafka](#)[LDAP](#)[Linux学习总结\(1\)](#)[Maven\(10\)](#)[Mina](#)[Mybatis\(8\)](#)[MyEclipse\(11\)](#)[MySQL\(2\)](#)[Nginx\(3\)](#)[Oracle学习使用总结](#)[PowerDesigner使用总结\(5\)](#)[redis](#)[RESTful架构\(1\)](#)[Servlet3.0\(4\)](#)[Snmp学习总结\(8\)](#)[Spring\(3\)](#)[SpringMVC学习总结](#)[SQLServer\(4\)](#)[Struts2\(4\)](#)[SVN](#)[VB.NET\(1\)](#)[WebLogic使用总结\(8\)](#)[WebService学习总结\(4\)](#)[WebSocket\(1\)](#)[WinForm学习总结](#)[XML学习总结\(2\)](#)[插件化开发](#)

```
11 * <p>ClassName: TestObjSerializeAndDeserialize<p>
12 * <p>Description: 测试对象的序列化和反序列<p>
13 * @author xudp
14 * @version 1.0 V
15 * @createTime 2014-6-9 下午03:17:25
16 */
17 public class TestObjSerializeAndDeserialize {
18
19     public static void main(String[] args) throws Exception {
20         SerializePerson(); //序列化Person对象
21         Person p = DeserializePerson(); //反序列Perons对象
22         System.out.println(MessageFormat.format("name={0}, age={1}, sex={2}",
23             p.getName(), p.getAge(), p.getSex()));
24     }
25
26     /**
27      * MethodName: SerializePerson
28      * Description: 序列化Person对象
29      * @author xudp
30      * @throws FileNotFoundException
31      * @throws IOException
32      */
33     private static void SerializePerson() throws FileNotFoundException,
34         IOException {
35         Person person = new Person();
36         person.setName("gacl");
37         person.setAge(25);
38         person.setSex("男");
39         // ObjectOutputStream 对象输出流, 将Person对象存储到E盘的Person.txt文件中, 完成对Person对象的序列化操作
40         ObjectOutputStream oo = new ObjectOutputStream(new FileOutputStream(
41             new File("E:/Person.txt")));
42         oo.writeObject(person);
43         System.out.println("Person对象序列化成功!");
44         oo.close();
45     }
46
47     /**
48      * MethodName: DeserializePerson
49      * Description: 反序列Perons对象
50      * @author xudp
51      * @return
52      * @throws Exception
53      * @throws IOException
54      */
55     private static Person DeserializePerson() throws Exception, IOException {
56         ObjectInputStream ois = new ObjectInputStream(new FileInputStream(
57             new File("E:/Person.txt")));
58         Person person = (Person) ois.readObject();
59         System.out.println("Person对象反序列化成功!");
60         return person;
61     }
62
63 }
```



代码运行结果如下：

[创业知识\(5\)](#)

[大数据/hadoop\(1\)](#)

[代码注释\(1\)](#)

[单点登录\(Yale CAS SSO\)](#)

[电脑基本技能](#)

[读书笔记](#)

[负载均衡\(1\)](#)

[互联网基础\(3\)](#)

[缓存框架](#)

[架构设计](#)

[框架整合\(1\)](#)

[敏捷开发\(1\)](#)

[权限设计](#)

[生活感悟\(3\)](#)

[数据库Sharding\(1\)](#)

[数据库理论基础\(3\)](#)

[微信开发\(3\)](#)

[我的开发框架](#)

[项目管理\(1\)](#)

[英语学习](#)

[原创小工具\(1\)](#)

[云计算\(2\)](#)

[哲学人生](#)

[职场感悟\(3\)](#)

[随笔档案](#)

[2016年4月 \(5\)](#)

[2016年3月 \(5\)](#)

[2016年2月 \(3\)](#)

[2016年1月 \(2\)](#)

[2015年7月 \(11\)](#)

[2015年3月 \(12\)](#)

[2015年2月 \(6\)](#)

[2015年1月 \(40\)](#)

[2014年12月 \(17\)](#)

[2014年11月 \(12\)](#)

[2014年10月 \(31\)](#)

[2014年8月 \(10\)](#)

[2014年7月 \(25\)](#)

[2014年6月 \(6\)](#)

[2014年5月 \(10\)](#)

[2014年4月 \(22\)](#)

[2014年3月 \(28\)](#)

[2014年2月 \(11\)](#)

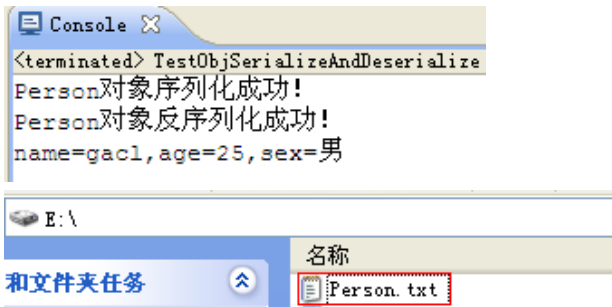
[2014年1月 \(2\)](#)

[2013年12月 \(16\)](#)

[2012年11月 \(1\)](#)

[最新评论](#)

[1. Re:JavaWeb学习总结\(五十\)——文件上传和下载](#)



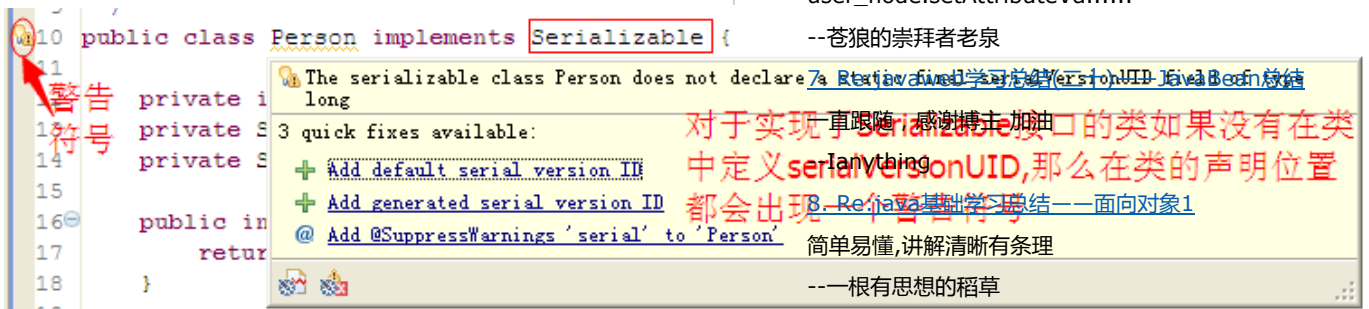
序列化Person成功后在E盘生成了一个Person.txt文件，而反序列化Person是读取E盘的Person.txt后生成了一个Person对象


### 三、serialVersionUID的作用

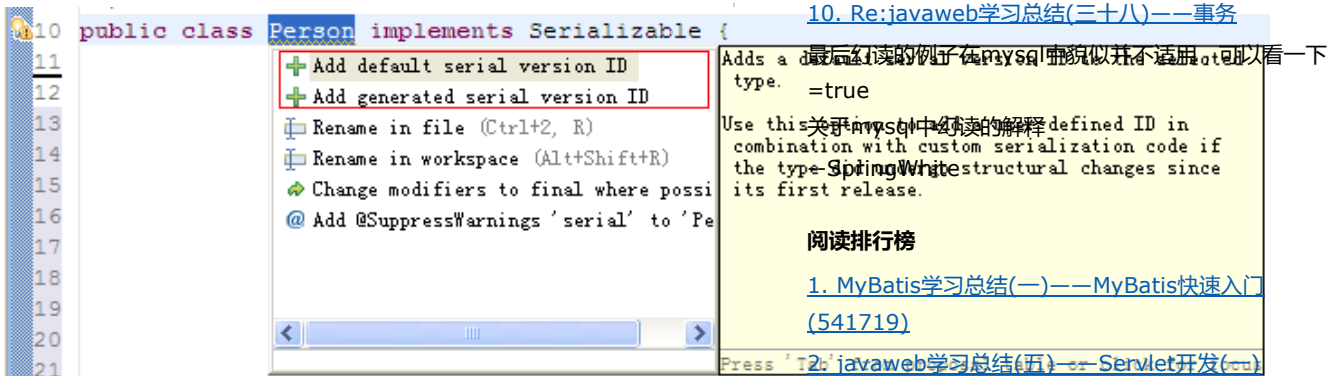
serialVersionUID: 字面意思是序列化的版本号，凡是实现Serializable接口的类都有一个表示序列化版本标识符的静态变量

```
1 private static final long serialVersionUID
```


实现Serializable接口的类如果类中没有添加serialVersionUID，那么就会出现如下的警告提示




用鼠标点击就会弹出生成serialVersionUID的对话框，如下图所示：



serialVersionUID有两种生成方式：

采用  Add default serial version ID 这种方式生成的serialVersionUID是1L，例如：

```
1 private static final long serialVersionUID = 1L;
```

采用  Add generated serial version ID 这种方式生成的serialVersionUID是根据类名，接口名，方法和属性等来生成

给力！

--漠北的仓鼠

[2. Re:MyBatis学习总结\(八\)——Mybatis3.x与Spring4.x整合](#)

很好

--frankchao1005

[3. Re:JNDI学习总结\(一\)——JNDI数据源的配置](#)

博主有没有，JNDI+GlassFish配置数据源的

--Q木偶人

[4. Re:JNDI学习总结\(一\)——JNDI数据源的配置](#)

@空城余梦love刚学，我也想知道这个...

--Q木偶人

[5. Re:MyBatis学习总结\(二\)——使用MyBatis对表执行CRUD操作](#)

@玄鉴你真是个人才，就服你...

--hacker浩浩

[6. Re:javaweb学习总结\(二十二\)——基于Servlet+JSP+JavaBean开发模式的用户登录注册](#)

我也遇到了注册不成功的问题，尝试解决。首先搜网上资料说是web.xml问题，一想基本上没用这个文件，排除。然后代码敲完，发现UserDaoImpl类中的user\_node.setAttributeVa.....

--苍狼的崇拜者老泉

[7. Re:javaweb学习总结\(二十一\)——JavaBean总结](#)

对于实现了Serializable接口的类如果没有在类中定义serialVersionUID,那么在类的声明位置都会出现一个警告符号

[8. Re:java基础学习总结——面向对象1](#)

简单易懂，讲解清晰有条理

--一根有思想的稻草

[9. Re:面试题收集——Java基础部分\(一\)](#)

第19面试题MVC中的分别简介的Controller少了一个"L"

--大红鹰再抹斜阳浓

[10. Re:javaweb学习总结\(三十八\)——事务](#)

最后例子的例子在mysql中似乎并不适用，可以看一下

阅读排行榜

[1. MyBatis学习总结\(一\)——MyBatis快速入门\(541719\)](#)

[2. javaweb学习总结\(五\)——Servlet开发\(372494\)](#)

[3. JavaWeb学习总结\(五十\)——文件上传和下载\(303231\)](#)

[4. MyEclipse使用总结——MyEclipse10安装SVN插件\(236321\)](#)

[5. JavaWeb学习总结\(一\)——JavaWeb开发入门\(235556\)](#)

[6. MyBatis学习总结\(五\)——实现关联表查询\(216245\)](#)

的，例如：

```
1 private static final long serialVersionUID = 4603642343377807741L;
```

添加了之后就不会出现那个警告提示了，如下所示：

```
public class Person implements Serializable {
    /**
     * 序列化版本号
     */
    private static final long serialVersionUID = 4603642343377807741L;
```

扯了那么多，那么serialVersionUID(序列化版本号)到底有什么用呢，我们用如下的例子来说明一下serialVersionUID的作用，看下面的代码：



```
1 import java.io.File;
2 import java.io.FileInputStream;
3 import java.io.FileNotFoundException;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.ObjectInputStream;
7 import java.io.ObjectOutputStream;
8 import java.io.Serializable;
9
10 public class TestSerialversionUID {
11
12     public static void main(String[] args) throws Exception {
13         SerializeCustomer();// 序列化Customer对象
14         Customer customer = DeserializeCustomer();// 反序列化Customer对象
15         System.out.println(customer);
16     }
17
18     /**
19      * MethodName: SerializeCustomer
20      * Description: 序列化Customer对象
21      * @author xudp
22      * @throws FileNotFoundException
23      * @throws IOException
24      */
25     private static void SerializeCustomer() throws FileNotFoundException,
26         IOException {
27         Customer customer = new Customer("gac1", 25);
28         // ObjectOutputStream 对象输出流
29         ObjectOutputStream oo = new ObjectOutputStream(new FileOutputStream(
30             new File("E:/Customer.txt")));
31         oo.writeObject(customer);
32         System.out.println("Customer对象序列化成功!");
33         oo.close();
34     }
35
36     /**
37      * MethodName: DeserializeCustomer
38      * Description: 反序列化Customer对象
39      * @author xudp
40      * @return
```

[7. JavaWeb学习总结\(十二\)——Session\(187224\)](#)  
[8. Spring常用注解\(180576\)](#)  
[9. Android开发学习总结\(一\)——搭建最新版本的Android开发环境\(167451\)](#)  
[10. javaweb学习总结\(十\)——HttpServletRequest对象\(一\)\(152858\)](#)

#### 评论排行榜

[1. JavaWeb学习总结\(五十\)——文件上传和下载\(163\)](#)  
[2. MyBatis学习总结\(一\)——MyBatis快速入门\(114\)](#)  
[3. javaweb学习总结\(二十二\)——基于Servlet+JSP+JavaBean开发模式的用户登录注册\(87\)](#)  
[4. javaweb学习总结\(五\)——Servlet开发\(一\)\(83\)](#)  
[5. MyBatis学习总结\(八\)——Mybatis3.x与Spring4.x整合\(71\)](#)  
[6. JavaWeb学习总结\(一\)——JavaWeb开发入门\(70\)](#)  
[7. MyBatis学习总结\(二\)——使用MyBatis对表执行CRUD操作\(70\)](#)  
[8. javaweb学习总结\(四\)——Http协议\(62\)](#)  
[9. MyBatis学习总结\(五\)——实现关联表查询\(60\)](#)  
[10. 谈谈对Spring IOC的理解\(57\)](#)

#### 推荐排行榜

[1. javaweb学习总结\(五\)——Servlet开发\(一\)\(184\)](#)  
[2. MyBatis学习总结\(一\)——MyBatis快速入门\(166\)](#)  
[3. JavaWeb学习总结\(一\)——JavaWeb开发入门\(136\)](#)  
[4. JavaWeb学习总结\(五十\)——文件上传和下载\(111\)](#)  
[5. 谈谈对Spring IOC的理解\(92\)](#)  
[6. javaweb学习总结\(二十二\)——基于Servlet+JSP+JavaBean开发模式的用户登录注册\(84\)](#)  
[7. MyBatis学习总结\(八\)——Mybatis3.x与Spring4.x整合\(81\)](#)  
[8. MyBatis学习总结\(五\)——实现关联表查询\(80\)](#)  
[9. Java基础学习总结——Java对象的序列化和反序列化\(79\)](#)  
[10. javaweb学习总结\(七\)——HttpServletResponse对象\(一\)\(79\)](#)

Powered by:

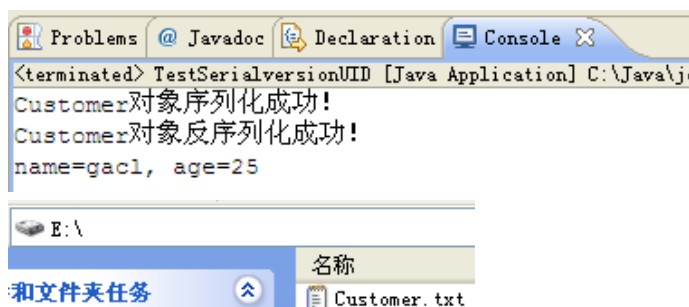
[博客园](#)

Copyright © 孤傲苍狼

```
41     * @throws Exception
42     * @throws IOException
43     */
44     private static Customer DeserializeCustomer() throws Exception, IOException {
45         ObjectInputStream ois = new ObjectInputStream(new FileInputStream(
46             new File("E:/Customer.txt")));
47         Customer customer = (Customer) ois.readObject();
48         System.out.println("Customer对象反序列化成功!");
49         return customer;
50     }
51 }
52
53 /**
54  * <p>ClassName: Customer<p>
55  * <p>Description: Customer实现了Serializable接口, 可以被序列化<p>
56  * @author xudp
57  * @version 1.0 V
58  * @createTime 2014-6-9 下午04:20:17
59  */
60 class Customer implements Serializable {
61     //Customer类中没有定义serialVersionUID
62     private String name;
63     private int age;
64
65     public Customer(String name, int age) {
66         this.name = name;
67         this.age = age;
68     }
69
70     /*
71     * @MethodName toString
72     * @Description 重写Object类的toString()方法
73     * @author xudp
74     * @return string
75     * @see java.lang.Object#toString()
76     */
77     @Override
78     public String toString() {
79         return "name=" + name + ", age=" + age;
80     }
81 }
```



运行结果：



序列化和反序列化都成功了。

下面我们修改一下Customer类，添加多一个sex属性，如下：





```

1 class Customer implements Serializable {
2     //Customer类中没有定义serialVersionUID
3     private String name;
4     private int age;
5
6     //新添加的sex属性
7     private String sex;
8
9     public Customer(String name, int age) {
10         this.name = name;
11         this.age = age;
12     }
13
14     public Customer(String name, int age, String sex) {
15         this.name = name;
16         this.age = age;
17         this.sex = sex;
18     }
19
20     /*
21      * @MethodName toString
22      * @Description 重写Object类的toString()方法
23      * @author xudp
24      * @return string
25      * @see java.lang.Object#toString()
26      */
27     @Override
28     public String toString() {
29         return "name=" + name + ", age=" + age;
30     }
31 }

```



然后执行反序列操作，此时就会抛出如下的异常信息：

```

1 Exception in thread "main" java.io.InvalidClassException: Customer;
2 local class incompatible:
3 stream classdesc serialVersionUID = -88175599799432325,
4 local class serialVersionUID = -5182532647273106745

```

意思就是说，文件流中的class和classpath中的class，也就是修改过后的class，不兼容了，处于安全机制考虑，程序抛出了错误，并且拒绝载入。那么如果我们真的有需求要在序列化后添加一个字段或者方法呢？应该怎么办？那就是自己去指定serialVersionUID。在TestSerialVersionUID例子中，没有指定Customer类的serialVersionUID的，那么java编译器会自动给这个class进行一个摘要算法，类似于指纹算法，只要这个文件多一个空格，得到的UID就会截然不同的，可以保证在这么多类中，这个编号是唯一的。所以，添加了一个字段后，由于没有显指定serialVersionUID，编译器又为我们生成了一个UID，当然和前面保存在文件中的那个不会一样了，于是就出现了2个序列化版本号不一致的错误。因此，只要我们自己指定了serialVersionUID，就可以在序列化后，去添加一个字段，或者方法，而不会影响到后期的还原，还原后的对象照样可以使用，而且还多了方法或者属性可以用。

下面继续修改Customer类，给Customer指定一个serialVersionUID，修改后的代码如下：



```
1 class Customer implements Serializable {
2     /**
3      * Customer类中定义的serialVersionUID(序列化版本号)
4      */
5     private static final long serialVersionUID = -5182532647273106745L;
6     private String name;
7     private int age;
8
9     //新添加的sex属性
10    //private String sex;
11
12    public Customer(String name, int age) {
13        this.name = name;
14        this.age = age;
15    }
16
17    /*public Customer(String name, int age,String sex) {
18        this.name = name;
19        this.age = age;
20        this.sex = sex;
21    }*/
22
23    /*
24     * @MethodName toString
25     * @Description 重写Object类的toString()方法
26     * @author xudp
27     * @return string
28     * @see java.lang.Object#toString()
29     */
30    @Override
31    public String toString() {
32        return "name=" + name + ", age=" + age;
33    }
34 }
```



重新执行序列化操作，将Customer对象序列化到本地硬盘的Customer.txt文件存储，然后修改Customer类，添加sex属性，修改后的Customer类代码如下：



```
1 class Customer implements Serializable {
2     /**
3      * Customer类中定义的serialVersionUID(序列化版本号)
4      */
5     private static final long serialVersionUID = -5182532647273106745L;
6     private String name;
7     private int age;
8
9     //新添加的sex属性
10    private String sex;
11 }
```



```
12     public Customer(String name, int age) {
13         this.name = name;
14         this.age = age;
15     }
16
17     public Customer(String name, int age, String sex) {
18         this.name = name;
19         this.age = age;
20         this.sex = sex;
21     }
22
23     /*
24      * @MethodName toString
25      * @Description 重写Object类的toString()方法
26      * @author xudp
27      * @return string
28      * @see java.lang.Object#toString()
29      */
30     @Override
31     public String toString() {
32         return "name=" + name + ", age=" + age;
33     }
34 }
```



执行反序列操作，这次就可以反序列成功了，如下所示：

```
Customer对象反序列化成功！
name=gac1, age=25
```

## 四、serialVersionUID的取值

serialVersionUID的取值是Java运行时环境根据类的内部细节自动生成的。如果对类的源代码作了修改，再重新编译，新生成的类文件的serialVersionUID的取值有可能也会发生变化。

类的serialVersionUID的默认值完全依赖于Java编译器的实现，对于同一个类，用不同的Java编译器编译，有可能会导导致不同的 serialVersionUID，也有可能相同。**为了提高serialVersionUID的独立性和确定性，强烈建议在一个可序列化类中显示的定义serialVersionUID，为它赋予明确的值。**

显式地定义serialVersionUID有两种用途：

- 1、在某些场合，希望类的不同版本对序列化兼容，因此需要确保类的不同版本具有相同的serialVersionUID；
- 2、在某些场合，不希望类的不同版本对序列化兼容，因此需要确保类的不同版本具有不同的serialVersionUID。

分类: [java基础总结](#)

标签: [java基础总结](#)

[好文要顶](#) [关注我](#) [收藏该文](#)



[孤傲苍狼](#)[关注 - 90](#)[粉丝 - 12469](#)[+加关注](#)

79

0

posted on 2014-06-09 16:55 [孤傲苍狼](#) 阅读(...) 评论(38) [编辑](#)  
[收藏](#)

## 评论

**#1楼 2014-06-09 23:08 微软一点都不软 \_**



好文

[支持\(2\)](#)[反对\(0\)](#)

**#2楼 2014-06-12 16:57 之奇一昂 \_**



谢楼主，讲得很仔细，学习了

[支持\(0\)](#)[反对\(0\)](#)

**#3楼 2014-07-30 09:41 伏草惟存 \_**



不错，讲的很易懂。

[支持\(0\)](#)[反对\(0\)](#)

**#4楼 2014-11-29 12:34 陶然楠轩 \_**



简单易懂

[支持\(0\)](#)[反对\(0\)](#)

**#5楼 2014-12-11 18:01 yly\_k \_**



给赞

[支持\(1\)](#)[反对\(0\)](#)

**#6楼 2015-01-29 22:31 巅峰寂寞 \_**



太棒了

[支持\(0\)](#)[反对\(0\)](#)

**#7楼 2015-01-29 23:58 江正军 \_**



记得N年前我还做过关于序列化与反序列化的总结：  
<http://jiangzhengjun.iteye.com/blog/519256>  
，在项目中曾经也用到过，不过很好，这个知识研究起来有点意思，涉及到二进制数据，不过按照 Effective Java，要少用

支持(1)反对(0)

**#8楼 2015-08-09 10:10 keep\_move \_**



赞！

支持(0)反对(0)

**#9楼 2015-08-10 14:50 yjie \_**



讲的很好，非常赞！

支持(0)反对(0)

**#10楼 2015-08-19 12:24 changxiangnan \_**



pickle和cPickle相当于java的序列化和反序列化操作

支持(0)反对(0)

**#11楼 2015-09-25 14:33 燃烧宇宙 \_**



赞

支持(0)反对(0)

**#12楼 2016-01-26 09:55 GGGGeek \_**



写的简直太好了！！！！

支持(0)反对(0)

**#13楼 2016-01-28 11:57 sstong123 \_**



好文章！

支持(0)反对(0)

**#14楼 2016-02-23 17:21 洪杰 \_**



必须赞

[支持\(0\)](#)[反对\(0\)](#)

**#15楼 2016-03-07 11:05 遁世千秋 \_**



好文，明白了

[支持\(0\)](#)[反对\(0\)](#)

**#16楼 2016-03-11 10:27 java\_lover \_**



写得很清楚，学习啦。

[支持\(0\)](#)[反对\(0\)](#)

**#17楼 2016-03-23 09:18 Theme \_**



谢谢楼主，正好不明白序列化的作用，现在明白了！

[支持\(0\)](#)[反对\(0\)](#)

**#18楼 2016-03-29 11:08 arronhu \_**



学习了，多谢分享！

[支持\(0\)](#)[反对\(0\)](#)

**#19楼 2016-04-03 22:56 带妳心菲 \_**



讲的很详细，谢谢楼主！

[支持\(0\)](#)[反对\(0\)](#)

**#20楼 2016-04-05 09:16 rojas \_**



文章很好，多问个问题 序列化和反序列化要什么注意的地方？

[支持\(1\)](#)[反对\(0\)](#)

**#21楼 2016-04-08 15:34 fairy1674 \_**



好东西,好东西!

[支持\(0\)](#)[反对\(0\)](#)

**#22楼 2016-04-11 22:17 fairy1674 \_**



博主的文章中有"然后执行反序列操作，此时就会抛出如下的异常信息："。并不是在所有的机器上都会报这个错误的。JDK中原话为"计算默认的 serialVersionUID 对类的详细信息具有较高的敏感性，根据编译器实现的不同可能千差万别，这样在反序列化过程中可能会导致意外的 InvalidClassException。"

[支持\(1\)](#)[反对\(0\)](#)

**#23楼 2016-04-20 10:28 jack\_ricky \_**



通俗易懂 谢谢

[支持\(0\)](#)[反对\(0\)](#)

**#24楼 2016-05-13 14:35 Mr.abner \_**



赞一个

[支持\(0\)](#)[反对\(0\)](#)

**#25楼 2016-05-14 08:32 Mr\_ Bo \_**



楼主能否讲一下应用场景，2个作用，不知道干嘛用的，什么情况下需要用序列化和反序列化？

[支持\(1\)](#)[反对\(0\)](#)

**#26楼 2016-06-17 09:16 琥珀子 \_**



谢谢楼主

[支持\(0\)](#)[反对\(0\)](#)

**#27楼 2016-06-29 10:47 nijian81 \_**



为了给楼主点个赞，特意注册了博客园的账号，写的真好！！

[支持\(0\)](#)[反对\(0\)](#)

**#28楼 2016-07-11 10:10 大风吹海 \_**



讲解的很仔细，很赞

[支持\(0\)](#)[反对\(0\)](#)

## #29楼 2016-08-04 10:38 孤星将夜 \_



谢谢~写得很清楚,初学者的我也能看懂

支持(0)反对(0)

## #30楼 2016-08-04 16:43 老王在你家 \_



写的很详细，对工作很有用的。楼主加油

支持(0)反对(0)

## #31楼 2016-08-18 16:30 wen73 \_



写的很好，详细步骤都有

支持(0)反对(0)

## #32楼 2016-09-16 11:37 yyzyyx \_



非常好，清晰明了，赞一个

支持(0)反对(0)

## #33楼 2016-09-21 15:32 v魂之挽歌 \_



我关注的是：评论咋加样式

支持(0)反对(0)

## #34楼 2016-09-26 16:20 小谢谢 \_



讲的很好。支持！

支持(0)反对(0)

## #35楼 2016-10-12 16:10 RoberTony \_



大赞博主，功德无量~！

支持(0)反对(0)

## #36楼 2016-10-13 17:53 quanhy \_



请求楼主帮忙，  
我修改了实体类，然后重新生成了一个  
serialVersionUID =



```
-199287337116318353L;  
当我发起工作事会出现报错：****  
serialVersionUID =  
1097123956217245034 , local class  
serialVersionUID =  
-5673743833880090083 ;
```

出现的这3个 serialVersionUID 都不一样；；即使我把实体类中serialVersionUID = -199287337116318353L;修改成 serialVersionUID = 1097123956217245034 ;也会报同样的错误；；这个这么解决

[支持\(0\)反对\(0\)](#)

**#37楼 2016-11-29 22:43 风的足迹212 \_**



说的太赞了，通俗易懂啊，例子很好

[支持\(0\)反对\(0\)](#)

**#38楼 2017-09-15 14:37 daBai\_**



想请问下 序列化和反序列化 一般作用场景。。。

一般是怎么使用的。。

[支持\(0\)反对\(0\)](#)

[刷新评论刷新页面返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#) 网站首页。

[【推荐】超50万VC++源码：大型工控、组态\仿真、建模CAD源码2018！](#)

[【腾讯云】小程序普惠节精美模板1元起](#)



**最新IT新闻：**

- [微信抄袭米聊？腾讯公关总监：呵呵 你真逗](#)
- [东芝考虑将存储芯片业务上市：如果不能卖掉的话](#)
- [干货分享：《欢乐坦克大战》微信小游戏开发总结](#)
- [2017年谷歌热搜旅游目的地榜单出炉，TOP1居然是这里！](#)

· [张小龙不谈情怀，那什么才是谈情怀的正确姿势？](#)

» [更多新闻...](#)

阿里云广告 banner，背景为深蓝色，带有白色和黄色的几何图形。左侧有阿里云 logo 和文字“阿里云”，中间是主要标题“告别高昂运维费用 云计算全面助力”，下方有副标题“40+款核心产品免费半年 再+8000津贴任意采购”，右侧有一个白色按钮，上面写着“立即申请”。

阿里云 告别高昂运维费用 云计算全面助力  
40+款核心产品免费半年 再+8000津贴任意采购 立即申请

最新知识库文章：

· [领域驱动设计在互联网业务开发中的实践](#)

· [步入云计算](#)

· [以操作系统的角度述说线程与进程](#)

· [软件测试转型之路](#)

· [门内门外看招聘](#)

» [更多知识库文章...](#)