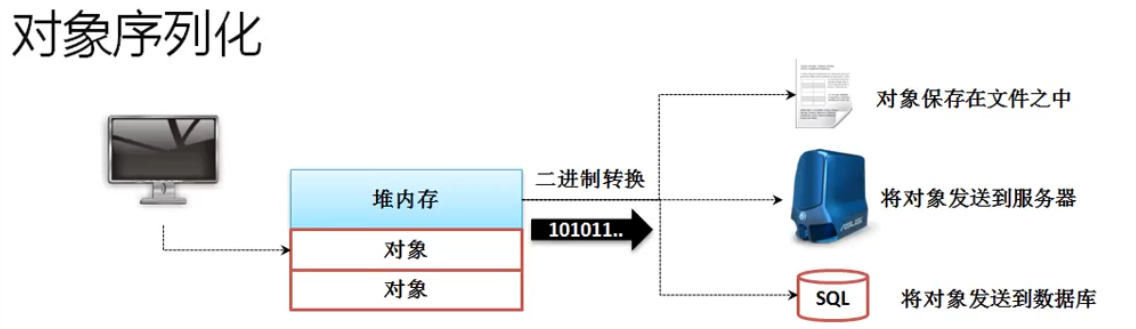
一、对象序列化

序列化的基本定义:

所谓的对象序列化指的是将内存中保存的对象以二进制数据流的形式进行处理，可以实现对象的保存或者网络传输。



然而并不是所有的对象都可以序列化，在Java中有一个强制性的要求:如果要序列化的对象，那么对象所在的类一定要实现java.io.Serializable父接口，作为序列化的标记，这个接口并没有任何的方法，也就是说他是描述的一种状态，java中还有一个Cloneable也是用来描述一种状态。

范例:定义一个可以序列化的对象

|  |
| --- |
|  |

此时Person类产生的每一个对象都可以实现二进制的数据传输，属于一个可以被序列化的程序类。

二、序列化与反序列化

|  |  |  |
| --- | --- | --- |
| 类名称 | 序列化:ObjectOutputStream | 反序列化:ObjectInputStream |
| 类定义 | public class ObjectOutputStream  extends [OutputStream](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/OutputStream.html)  implements [ObjectOutput](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/ObjectOutput.html), [ObjectStreamConstants](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/ObjectStreamConstants.html) | public class ObjectInputStream  extends [InputStream](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/InputStream.html)  implements [ObjectInput](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/ObjectInput.html), [ObjectStreamConstants](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/ObjectStreamConstants.html) |
| 构造方法 | protected ObjectOutputStream([OutputStream](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/OutputStream.html) out) throws [IOException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/IOException.html), [SecurityException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/SecurityException.html) | protected ObjectInputStream([InputStream](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/InputStream.html) in) throws [IOException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/IOException.html), [SecurityException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/SecurityException.html) |
| 操作方法 | public final void writeObject​([Object](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/Object.html) obj) throws [IOException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/IOException.html) | public final [Object](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/Object.html)  readObject() throws [IOException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/IOException.html), [ClassNotFoundException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/ClassNotFoundException.html) |

范例:序列化与反序列化Person对象

|  |
| --- |
| package com.company.对象序列化;  import java.io.\*;  public class ObjectDemo {  private static final File SAVE\_FILE = new File("G:"+File.separator+"mldn.person");  public static void main(String[] args) throws Exception {  //saveObject(new Person("王五",12));  System.out.println(loadObject());  }  public static void saveObject(Object obj) throws Exception{  ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(SAVE\_FILE));  oos.writeObject(obj);  oos.close();  System.out.println("成功序列化");  }  public static Object loadObject() throws Exception{  ObjectInputStream ois = new ObjectInputStream(new FileInputStream(SAVE\_FILE));  Object object = ois.readObject();  ois.close();  return object;  }  }  class Person implements Serializable{  private String name ;  private int age ;  public Person(String name ,int age ){  this.name = name ;  this.age = age ;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  @Override  public String toString() {  return "Person{" +  "name='" + name + '\'' +  ", age=" + age +  '}';  }  } |

在Java中的对象序列化与反序列化必须使用内部提供的对象操作流，因为这里牵扯到一个二进制的处理格式，因此我们不能自定义处理，另外如果想要实现一组对象的序列化可以使用对象数组完成。

三、transient关键字

默认情况下当执行了对象序列化的时候会将类中的全部属性进行序列化，但是很多情况下一些属性不需要序列化处理，此时就可以使用transient关键字定义属性了。

·private transient String name;

在进行序列化处理的时候name 属性的内容是不会被保存下来的，换言之，读取数据时name 将会是其对应的默认值（null）；

如果类之中有一些需要计算保存的属性，就可以使用transient关键字。