涉及对象流。

1、什么是序列化与反序列化？

序列化：指把堆内存中的 Java 对象数据，通过某种方式把对象存储到磁盘文件中或者传递给其他网络节点（在网络上传输）。这个过程称为序列化。通俗来说就是将数据结构或对象转换成二进制串的过程。

反序列化：把磁盘文件中的对象数据或者把网络节点上的对象数据，恢复成Java对象模型的过程。也就是将在序列化过程中所生成的二进制串转换成数据结构或者对象的过程。

2、为什么要做序列化？

①、在分布式系统中，此时需要把对象在网络上传输，就得把对象数据转换为二进制形式，需要共享的数据的 JavaBean 对象，都得做序列化。

②、服务器钝化：如果服务器发现某些对象好久没活动了，那么服务器就会把这些内存中的对象持久化在本地磁盘文件中（Java对象转换为二进制文件）；如果服务器发现某些对象需要活动时，先去内存中寻找，找不到再去磁盘文件中反序列化我们的对象数据，恢复成 Java 对象。这样能节省服务器内存。

3、Java 怎么进行序列化？

①、需要做序列化的对象的类，必须实现序列化接口：Java.lang.Serializable 接口（这是一个标志接口，没有任何抽象方法），Java 中大多数类都实现了该接口，比如：String，Integer

②、底层会判断，如果当前对象是 Serializable 的实例，才允许做序列化，Java对象 instanceof Serializable 来判断。

③、在 Java 中使用对象流来完成序列化和反序列化。

ObjectOutputStream:通过 writeObject()方法做序列化操作

ObjectInputStream:通过 readObject() 方法做反序列化操作

|  |
| --- |
| public class TestObjectStream {  public static File container;  static {  container = new File("io"+File.separator+"serd"+File.separator+"people.txt");  if(!container.exists()){  try {  container.createNewFile();  } catch (IOException e) {  e.printStackTrace();  }  }  }  public static void main(String[] args) throws IOException, ClassNotFoundException {  // serializePeopleToFile();  // dederializePeopleFromFile();  serializePeopleToMemoryAndDesrializeFromMemory();  }  /\*将对象序列化到文件中\*/  public static void serializePeopleToFile() throws IOException {  OutputStream out = new FileOutputStream(container);  ObjectOutputStream outputStream = new ObjectOutputStream(out);  outputStream.writeObject(new People(20,"csc"));  outputStream.close();  }  /\*将对象从文件中反序列化出来\*/  public static void dederializePeopleFromFile() throws IOException, ClassNotFoundException {  InputStream input= new FileInputStream(container);  ObjectInputStream objectInputStream = new ObjectInputStream(input);  People people = (People) objectInputStream.readObject();  objectInputStream.close();  System.out.println(people.toString());  }  /\*将对象序列化到内存中，并读出来\*/  public static void serializePeopleToMemoryAndDesrializeFromMemory() throws IOException, ClassNotFoundException {  ByteArrayOutputStream out = new ByteArrayOutputStream();  ObjectOutputStream outputStream = new ObjectOutputStream(out);  outputStream.writeObject(new People(11,"haha"));  InputStream input = new ByteArrayInputStream(out.toByteArray());  ObjectInputStream inputStream = new ObjectInputStream(input);  People people = (People) inputStream.readObject();  System.out.println(people.toString());  inputStream.close();  outputStream.close();  }  }  class People implements Serializable{  private int age;  private String name;  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public People(int age, String name) {  this.age = age;  this.name = name;  }  @Override  public String toString() {  return "People{" +  "age=" + age +  ", name='" + name + '\'' +  '}';  }  } |