RMI规范--第五章服务器接口 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/145/2021_2022_RMI_E8_A7_84_E8_8C_83-_c104_145019.htm java.rmi.server 包包含通常用于实现远程对象的接口与类。主题：RemoteObject 类RemoteServer 类UnicastRemoteObject 类Unreferenced 接口RMISecurityManager 类RMIClassLoader 类LoaderHandler 接口RMI 套接字工厂RMIFailureHandler 接口LogStream 类stub 和 skeleton 编译器5.1 RemoteObject 类类 java.rmi.server.RemoteObject 将 java.lang.Object 行为实现于远程对象。实现方法 hashCode 和 equals 将允许将远程对象引用存储在散列表中进行比较。如果两个 Remote 对象引用同一个远程对象，则方法 equals 的返回值为 true。它负责比较远程对象的远程对象引用。方法 toString 返回一个说明远程对象的字符串。该字符串的内容和语法与实现有关且可变
。java.lang.Object 中的其它方法保留了它们的原始实现
。package java.rmi.server.public abstract class RemoteObjectimplements java.rmi.Remote, java.io.Serializable{protected transient RemoteRef ref.protected RemoteObject().protected RemoteObject(RemoteRef ref).public RemoteRef getRef().public static Remote toStub(java.rmi.Remote obj)throws java.rmi.NoSuchObjectException.public int hashCode().public boolean equals(Object obj).public String toString().}因为 RemoteObject 是抽象类，所以无法实例化。因此，RemoteObject 的构造函数必须从子类实现中调用。第一个 RemoteObject 构造函数将创建带空的远程引用的

RemoteObject。第二个 RemoteObject 构造函数将创建带给定远程引用 ref 的 RemoteObject。方法 getRef 返回该远程对象的远程引用。方法 toStub 返回一个远程对象 obj 的 stub 并作为参数传送。该操作仅在已经导出远程对象实现后才有效。如果找不到远程对象的 stub，该方法就抛出NoSuchObjectException。5.1.1 RemoteObject 类覆盖的对象方法java.lang.Object 类中用于方法 equals、hashCode 和 toString 的缺省实现不适用于远程对象。因此，RemoteObject 类提供了这些方法在语义上更合适于远程对象的实现。equals 和 hashCode 方法为将远程对象用作散列表中的主键，我们必须在远程对象实现中覆盖 equals 和hashCode 方法，这些方法是由类 java.rmi.server.RemoteObject 覆盖的：java.rmi.server.RemoteObject 类实现 equals 方法决定了两个对象的引用是否相等，而不是两个对象的内容是否相等。这是因为决定内容是否相等时需要远程方法调用，而 equals 的签名不允许抛出远程异常。对于所有引用同一底层远程对象的远程引用，java.rmi.server.RemoteObject类实现的 hashCode 方法返回同一个值（因为对相同对象的引用被认为是相等的）。toString 方法toString 方法被定义为返回表示对象的远程引用的字符串。字符串的内容视引用的类型而定。单体（单路传送）对象的当前实现一个对象标识符以及与传输层有关的该对象的其他信息（例如主机名和端口号）。clone 方法只有在对象支持 java.lang.Cloneable 接口时才能用 Java 语言的缺省机制来复制。由 rmic 编译器生成的远程对象的 stub 将被声明为终态，且不实现Cloneable 接口，因此无法复制 stub。5.1.2 序列化形式RemoteObject 类实现专门的（私用）方法 writeObject

和方法 readObject，它们由对象序列化机制调用来处理向 java.io.ObjectOutputStream 中序列化数据。RemoteObject 的序列化形式由下列方法写入：private void writeObject(java.io.ObjectOutputStream out)throws java.io.IOException, java.lang.ClassNotFoundException.如果 RemoteObject 的远程引用域 ref 为空，则该方法抛出java.rmi.MarshalException。如果远程引用 ref 为非空：ref 的类通过调用其 getRefClass 方法来获得，该方法通常返回远程引用类的非打包全名。如果返回的类名为非空：ref 的类名将以 UTF 格式写到流 out 中。调用 ref 的方法 writeExternal，传递的参数为流 out，从而使 ref 可以将其外部表示法写到流中。如果 ref.getRefClass 返回的类名为空：则将一个 UTF 格式的空字符串写到流 out 中。ref 被序列化到流 out（即利用 writeObject）。序列化恢复时，RemoteObject 的状态将由 ObjectInputStream 调用该方法利用其序列化形式进行重构：private void readObject(java.io.ObjectInputStream in)throws java.io.IOException, java.lang.ClassNotFoundException.首先，ref 的类名（UTF 字符串）将从流 in 中读出。如果类名为空字符串：则从流中读出对象，然后将 ref 初始化为该对象（即通过调用 in.readObject）如果类名为非空：则 ref 的完整类名由字符串 java.rmi.server.RemoteRef.packagePrefix 的值和“.”加上从流中读取的类名相连接而成。创建 ref 类的实例（利用上述完整类名）。该新实例（成为 ref 域）从流 in 中读取其外部形式。5.2 RemoteServer 类java.rmi.server.RemoteServer 类是服务器实现类java.rmi.server.UnicastRemoteObject 