链滴

Parcelable&Serializable 序列化使用笔记

作者：pencilso

原文链接：https://ld246.com/article/1493028556883

来源网站：链滴

Parcelable序列化接口使用##

首先先实现Parcelable接口 以下是JavaBean的代码片段

```
import android.os.Parcel;
import android.os.Parcelable;
public class UserBean implements Parcelable {
    private int id;
    private String name;
    private int age;

    public UserBean(int id, String name, int age) {
        super();
        this.id = id;
        this.name = name;
        this.age = age;
    }

    /**
     * 该方法为Parcelable接口中的抽象方法 这里的返回值 返回0即可
     */
    @Override
    public int describeContents() {
        return 0;
    }

    /**
     * 该方法为Parcelable接口中的抽象方法
     *
     * @param dest
     *            该对象为序列的操作对象 通过该对象 进行序列化属性
     */
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeInt(id); // 序列int类型
        dest.writeString(name); // 序列String类型
        dest.writeInt(age);
        // --还有double等等类型 writeDouble .....
    }

    /**
     * 添加一个反序列化属性的构造方法
     *
     * @param source
     */
    private UserBean(Parcel source) {
        /**
         * 反序列化顺序 跟序列化的顺序一致
         */
        id = source.readInt(); // 反序列化int类型
        name = source.readString();// 反序列化String类型
        age = source.readInt();
```

```
    }

    /**
     * 实现Parcelable接口下的创建类
     */
    public static final Parcelable.Creator CREATOR = new Parcelable.Creator() {
        /**
         * 根据size大小 创建一个对象数组
         */
        @Override
        public UserBean[] newArray(int size) {
            return new UserBean[size];
        }

        @Override
        public UserBean createFromParcel(Parcel source) {
            return new UserBean(source);
        }
    };
}

/**使用Intent传递
/--传递方:
    UserBean ub = new UserBean();
    Intent intent = new UserBean(1, "张三", 19);
    intent.putExtra("user", ub);
    startActivity(intent);
/--接收方:
    UserBean ub = getIntent().getParcelableExtra("user");
```

Serializable序列化接口使用

相对来说 Java的Serializable接口使用要简单的多 并且Serializable是可以序列化到本地的而Parcelable则不支持

代码片段

```
import java.io.Serializable;
public class UserBean implements Serializable {
    private static final long serialVersionUID = -1885454655185328146L;
    private int id;
    private String name;
    private int age;

    public UserBean(int id, String name, int age) {
        super();
        this.id = id;
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
```

```
        return "UserBean [id=" + id + ", name=" + name + ", age=" + age + "]";
    }

}

/**使用Intent传递
/--传递方:
    UserBean ub = new UserBean();
    Intent intent = new UserBean(1, "张三", 19);
    intent.putExtra("user", ub);
    startActivity(intent);
/--接收方:
    UserBean ub = (UserBean) getIntent().getSerializableExtra("user");
```

##Serializable序列化到本地与反序列化##

```
    /**
     * 输出到本地文件当中
     */
    ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("file"));
    out.writeObject(obj);
    out.close();//一般close应该写在finally代码块中的
    /**
    * 从本地文件当中  反序列化成对象
    */
    ObjectInputStream in = new ObjectInputStream(new FileInputStream("path"));
    Object obj = in.readObject();
    in.close();
```

##Serializable使用Socket序列化输出与反序列化##

以前刚学Java的时候玩过socket，当时还不知道Json，于是就拿Serializable对象来传输数据了，记一下用法。

```
    /**
     * 使用Socket输出Serializable
     */
    ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
    out.close();
    out.writeObject(obj);

    /**
     * 从socket反序列化推过来的Serializable
     */
    ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
    Object obj = in.readObject();
```