Last Updated: January 24, 2018 · itseranga 

Passing objects between activities in android

ANDROID SERIALIZATION PARCELABLE

Passing data with intent

- Passing simple data types (String, int, double,…ect) between activities is easy. We can just put the them to intent with unique key and send it to an another activity.
- But it is bit complex when passing custom objects between activities. This is the place where `serialization` comes.

What is Serialization?

- Serialization is converting data from a fast, efficient, internal representation to something that can be kept in a persistent store or transmitted over a network.
- Converting data to its serialized form is often called `marshaling` it. Converting it back to its live, in-memory representation is called deserializing or `unmarshaling`.

More about serialization - https://www.inkling.com/read/programming-android-mednieks-1st/chapter-6/serialization

- In android there are two ways to achieve `marshaling` and `unmarshaling` of java objects

1. Serializable (Implment object as Serializable)
2. Parcelable (Implement object as Parcelable)

Serializable vs Parcelable

- Serializable is a marker interface, which implies the user cannot marshal the data according to their requirements. So when object implements `Serializable` Java will automatically serialize it.
- Parcelable is android own serialization protocol. In Parcelable, developers write custom code for marshaling and unmarshaling. So it creates less garbage objects in comparison to Serialization
- The performance of Parcelable is very high when comparing to Serializable because of its custom implementation
- It is highly recommended to use Parcelable implantation when serializing objects in android

Parcelable objects

- When need to pass custom object between activities we can mark the object as Parcelable(Implement the Parcelable interface). Then object capable to put in to an intent with `intent.putExtra` method
- Following is an example implementation of parcelable object(`User`)

https://github.com/erangaeb/dev-notes/blob/master/android-parcelable/User.java

- This is a simple object which keeps user related data. `User` object implements as Parcelable.
- There are two methods to override from `Parcelable` interface `describeContents` and `writeToParcel`. Actual object serialization do in `writeToParcel` method.

```
/**
 * Define the kind of object that you gonna parcel,
 * You can use hashCode() here
 */
@Override
public int describeContents() {
    return 0;
}
```

```
/**
 * Actual object serialization happens here, Write object content
```

```
     * to parcel, reading should be done according to this write order
     * param dest - parcel
     * param flags - Additional flags about how the object should be written
     */
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(id);
        dest.writeString(username);
        dest.writeString(email);
        dest.writeString(password);
    }
```

- Object need to have **`Parcelable.Creator`**. This field is needed for Android to be able to create new objects, individually or as arrays

```
    /**
     * This field is needed for Android to be able to
     * create new objects, individually or as arrays
     *
     * If you don't do that, Android framework will raises an exception
     * Parcelable protocol requires a Parcelable.Creator object
     * called CREATOR
     */
    public static final Parcelable.Creator<User> CREATOR = new Parcelable.Creator<User>

        public User createFromParcel(Parcel in) {
            return new User(in);
        }

        public User[] newArray(int size) {
            return new User[size];
        }
    };
```

- This User object is a simple parcelabel object, There can be some more complex objects, for an instance you have an object that references another object. In this kind of scenario they would both need to be Parcelable
- Following is an example of more complex parcelable object(**`Sensor`**)

https://github.com/erangaeb/dev-notes/blob/master/android-parcelable/Sensor.java

- `Sensor` object contains few special fields

```
1. User object(user)
2. User list(sharedUsers)
3. boolean field(isMySensor)
```

- When writing `user` object of sensor to the parcel, it uses

```
dest.writeParcelable(user, flags);
```

- When reading `user` object from parcel it needs class loader

```
this.user = in.readParcelable(User.class.getClassLoader());
```

- When writing user list(`sharedUsers`) to the parcel, it uses

```
dest.writeTypedList(sharedUsers);
```

- When reading user list(`sharedUsers`) from parcel it needs `User.CREATOR` instance

```
this.sharedUsers = new ArrayList<User>();
in.readTypedList(sharedUsers, User.CREATOR);
```

- Instead of writing boolean value to parcel, we convert it to an integer value and save it as an integer(since no method to write booleans directly to the parcel)

```
dest.writeInt(isMySensor ? 1 : 0);
```

- When reading the boolean value, we get the boolean value from the saved integer

```
this.isMySensor = (in.readInt() != 0);
```

Pass the object with intent

- In starting activity we can set the parcelable object to an intent

```
User user = new User("1", "u", "u", "u");
ArrayList<User> userList = new ArrayList<User>();
Sensor sensor = new Sensor("1", "s", "s", true, user, userList);
intent.putExtra("com.score.senzors.pojos.Sensor", sensor);
```

- In destination activity we can get the parcelable object from intent extras(bundle)

```
Bundle bundle = getIntent().getExtras();
Sensor sensor = bundle.getParcelable("com.score.senzors.pojos.Sensor")
```

Written by eranga bandara

5 Responses

noxiouswinter

Parcelable looks like the best way to do it. Serializable the easiest. But there is no escaping having to add the objects as extras to the intent using keys and getting it back from the other end etc.

I had always wondered why this can't be as simple as calling into a method of the other activity. I recently wrote a utility library that makes it almost as simple as that. You can check it out here(https://github.com/noxiouswinter/gnlib_android/wiki#gnlauncher).

I have not profiled this yet but it may or may not be the most efficient approach. Feel free to contribute to the project if you can think of ways to improve it.

over 1 year ago ·

manasouza

When I have to transfer some custom object which on its content there're other objects that are from a third-party integration (so, I cant make them implement Serializable or Parcelable for example), this approach cannot happen. There's another way out?

over 1 year ago ·

nemanjakovac

There is a cool library, Parceler, that can help you with third party objects not implementing Parcelable, but also with your classes. However it has its downfalls - for more info see http://nemanjakovacevic.net/blog/english/2015/03/24/yet-another-post-on-serializable-vs-parcelable/

over 1 year ago ·

asifmujteba

Parcelable or Serializable, In both ways we have to add extra code and still get performance and memory hit. especially when we start passing more complex objects this is can become a serious performance hit.

If we look in the Android Docs then there is another way of passing objects which is much faster. I have created a simple yet powerful class for this purpose (https://github.com/asifmujteba/AndroidObjectStore), Please have a look at it and let me know! Thanks.

over 1 year ago ·

DarthWendigo

Is a good practice setting the extra as JSON String?

over 1 year ago ·

Filed Under

Android Development Tips

Sponsors

Awesome Job

See All Jobs →

Senior JavaScript Software Engineer (EU)
Keemotion · Belgium, Brussels, Louvain-La-Neuve · Full Time