Difference between serializable and parcelable in android

I'm not robot
reCAPTCHA

Continue

Often, when we develop applications, we have to transfer data from one action to another. Of course, we can't do it directly. The data we want to transmit must be included in the relevant object of intent. And if we want to move a complex POJO (such as a person, car, employee, etc.), we also need to perform some additional steps to make this object suitable for transmission. To do this, our object must be either Serializable or Parcelable.What serializable? Serializable is a standard Java interface. It is not part of Android SDK. It's simplicity that's beauty. Just by implementing this interface, your POJO will be ready to move from one action to another. In the next piece of code, you can see how easy it is to use this interface. Because Serializable is a token interface, we don't need to introduce tons of additional methods. When we mark our POJO with it, Java will try to serialize it. Of course, this simple approach has a price. Reflection is used in the process and many additional objects are created along the way. This can lead to a lot of garbage. The result is poor performance and battery leakage. What is Parcelable? Parcelable is another interface. Although it's a rival (Serializable in case you forget), it's part of Android SDK. Now, Parcelable has been specially designed in a way that there is no reflection when using it. That's because, we're being really explicit for the serialization process. In the code snippet below you can see an example of using a parcelable interface: Of course, there is a price we have to pay when using Parcelable as well! Because of the boiler code, it's much harder to maintain and understand POJO from above. Parcelable VS SerializableIf you are looking online for information on this topic, you will most likely come to the conclusion that there will be an absolute winner for this comparison. There are people and articles out there that support either one or the other approach. So I'll introduce you to both sides and leave you to decide for yourself! The first team is behind the idea that Parcelable is much faster and better than Serializable. Of course, there is the data behind this statement. Tests conducted by Philip Breo show that Parcelable is more than 10 times faster than Serializable. Some other Google engineers are behind this statement as well. You can find a link to Philip's article in the Links section below. Now, the second team claims that we are all doing it wrong! And their arguments sound reasonable enough! According to them, the default Serializable approach is slower than Parcelable. And here we have an agreement between the two sides! BUT, it's unfair to compare these two at all! Because with Parcelable we're at the very write custom code. Code Code Created for this one POJO. Thus, garbage is not created and the results are better. But with Serializable's default approach, we rely on the process of automatic Java serialization. The process doesn't seem to be custom at all and creates a lot of garbage! So the worst results. Now there is a different approach. The entire automatic process behind Serializable can be replaced by a user code that uses writeObject and readObject (methods). These methods are specific. If we want to rely on a serial approach combined with custom serialization behavior, then we need to include these two methods with the same exact signature as below: Strange! Weird! Unusual! But here's how it works! Now, in these two methods, we can incorporate our custom logic. If done correctly, the garbage associated with the default serial approach will no longer be a factor! And now the comparison between Parcelable and Custom Serializable seems fair! The results can be amazing! Serializable's custom approach is more than 3 times faster for writing and 1.6 times faster to read than Parcelable. You can find the BitBucket project with test data in the Resources section to play. In my opinion, the differences in speed between the two approaches will be almost minor in most cases. So in the end, it's much more important to get the job done and have happy users than to have the app running with 0.000042 milliseconds faster. Links: Check out my personal blog articles related to Android! If you liked it, click ♥ below to let other people see it here on Medium. Photo Victor Aznabayev on Unsplash: When you develop an android Parcelable vs Serializable app, you often add data about your intention to transfer data from one action to another. What is Serializable? Serializable is an interface in standard Java, not Android SDK, and the objects in the class that implemented this interface are now ready to move from one action to another. You can see how easy it is to use this interface in the following code: Serializable is a simple marker interface that only tells you that the class is serialized and has no methods, so it's very easy to use by users. Serializable uses reflection internally to process serialization. Reflection is a process It is used during the process and generates many additional objects. Many of these waste scum are targeted by garbage collectors, resulting in poor performance and battery consumption due to excessive garbage collector behavior. What is Parcelable? Parcelable is another interface for serialization. Unlike Serializable, it's an Android SDK interface, not a standard Java. Parcelable is designed to disable the reflection. Unlike Serializable, you don't need reflection for automatic processing because you explicitly write serialization techniques. This makes it difficult to understand classes and makes it difficult to add new features. Serializable If this system is worth it, Parcelable should pay for the efforts of users who implement, support and support. Parcelable vs. Serializable On the Internet you will find a lot of information comparing Parcelable to Serializable. And it's up to you to judge the results. Of course, there is data to support this assertion. Philipe Breault's experimental results show that Parcelable is 10 times faster than Serializable. Some Google engineers also support this result, with reference to Philipe Breault's post in the Notes section at the bottom of this article. And this statement seems reliable enough. Serializable' on basic use is slower than Parcelable. There is no disagreement between the two groups, but Philipe Breault testing methods are unfair. Parcelable, as described above, requires you to write a special user code for only one class object. Parcelable serializable with custom code There is no reason to create other garbage, and the result is better performance. This process is certainly not as complicated as Parcelable, but it produces a lot of garbage in the process. So you get the worst results. Serialized can be automatically processed during the serialization process and replaced with writeObject () and readObject () methods implemented by the user. To use the Serializable approach with the serialization method you've implemented, you need to implement the following methods, as if you were using Parcelable, writeObject, and readObject () Serializable methods may include processing logic for that class. If done correctly, the debris generated by the serializable basic use method is no longer generated. The test results will be a blow. Serializable, which implements a certain class processing logic, writes more than parcels and is 1.6 times faster to read. But if you test serializable in the same way as Parcelable, you can see that it is not. What do you want to focus on? Or is it speed? If the speed is fast, would you like to use Serializable, which implemented a certain class processing logic instead of Parcelable? Or would you like to continue using Parcelable? what is the difference between serializable and parcelable which is the best approach in android

normal_5f87156311598.pdf
normal_5f87195dcd64c.pdf
normal_5f86f62be3a64.pdf
normal_5f86f630e6a1d.pdf
normal_5f870f1635d67.pdf
salo full movie english subtitles
plan anual trabajo comision tecnico
glencoe algebra 2 study guide and intervention answer key chapter 7

download video masha and the bear te
physics kinetic energy worksheet
rose gold party decorations
limited warranty deed south carolina
taylor error analysis pdf
ozark trail 14x12 cabin dome tent instructions
realm grinder eligible artifacts
league of legends twitch jungle guide
after 2 libro pdf italiano
straight to hell book pdf
mossberg 500 owners manual
normal_5f86f9dae2a0c.pdf
normal_5f86fea170fd7.pdf