Use of Parceler with Kotlin data class with constructor for serialization

Asked 9 years ago Modified 5 years ago Viewed 12k times

23

Is there a way to use Parceler with Kotlin data classes and constructor for serialization without using `@ParcelProperty` annotation for each field?

If I try and use library like this:

```
@Parcel
data class Valve @ParcelConstructor constructor(val size: Int)
```

I get `Error:Parceler: No corresponding property found for constructor parameter arg0`. But if I add `@ParcelProperty("size")` it works just fine.
Why is that?

Update:
There are other another way to use this library.
I could just remove `@ParcelConstructor` annotation, but then I will get error `Error:Parceler: No @ParcelConstructor annotated constructor and no default empty bean constructor found.`
I think (haven't tested it) I also could make all constructor parameters optional and add `@JvmOverloads` but that has a side effect that I have to check all properties of the class if they are null or not.

Update 2:
This is what worked for me:

```
@Parcel
data class Valve(val size: Int? = null)
```

In short generated Java class must have default empty constructor. One way to achieve that is to do as above - all variables should have default values.

parcelable kotlin parceler

Share
Improve this question
Follow

edited Jan 12, 2016 at 20:27

asked Nov 24, 2015 at 10:59

Martynas Jurkus
9,301 ● 13 ● 61 ● 102

4 Answers

Sorted by: Highest score (default)

29

According to the docs, Parceler by default works with public fields. But a usual Kotlin `data class` (as in your example) is rather a "traditional getter/setter bean", since every Kotlin property is represented by a private field and a getter/[setter].

TL; DR: I think this will work:

```
@Parcel(Serialization.BEAN)
data class Valve(val size: Int = 10)
```

Note the default value, it allows Kotlin to automatically generate an additional empty constructor, which is required by the Java Been specification.

Another way would be to mark the constructor that we already have:

```
@Parcel(Serialization.BEAN)
data class Driver @ParcelConstructor constructor(val name: String)
```

The specific document: https://github.com/johncarl81/parceler#gettersetter-serialization

Share

Improve this answer

Follow

edited Nov 24, 2015 at 15:06

answered Nov 24, 2015 at 11:28

voddan

33.7k ● 9 ● 81 ● 89

That wouldn't work since it requires default empty bean constructor. I'll update my question. – Martynas Jurkus Nov 24, 2015 at 11:51

to have an empty constructor you can provide a default value to `size`. Updated – voddan Nov 24, 2015 at 11:54

1 btw, for having an empty constructor you should not need `@JvmOverloads`, what I said above should be sufficient. Let me know when you try it out – voddan Nov 24, 2015 at 12:15

1 You may ask @MartynasJurkus, he made it work (obviously). For that you may want to post the compilation error you are getting. Or create a separate question, since chances are you problem is build-related, not kotlin-related. – voddan Jan 12, 2016 at 16:27

1 @FábioCarballo see my updated question on how I got it working based on this answer – Martynas Jurkus Jan 12, 2016 at 20:29

I know this question already has an answer, but for future viewers who are also struggling to get Parceler to work with kotlin data objects, I wrote a new annotation

10

processor to generate the Parcelable boilerplate for Kotlin data classes. It's designed to massively reduce the boilerplate code in making your data classes Parcelable:

https://github.com/grandstaish/paperparcel

Usage:

Annotate your data class with `@PaperParcel`, implement `PaperParcelable`, and add a JVM static instance of the generated `CREATOR` e.g.:

```
@PaperParcel
data class Example(
  val test: Int,
  ...
) : PaperParcelable {
  companion object {
    @JvmField val CREATOR = PaperParcelExample.CREATOR
  }
}
```

Now your data class is `Parcelable` and can be passed directly to a `Bundle` or `Intent`

Edit: Update with latest API

Share
Improve this answer
Follow

edited Nov 27, 2016 at 8:35

answered Feb 3, 2016 at 6:29

Bradley Campbell
9,788 ● 7 ● 38 ● 47

5

Just add the default constructor:

```
@Parcel
data class Valve(val size: Int) {
    constructor() : this(0)
}
```

Share Improve this answer Follow

answered Feb 26, 2016 at 0:47

dmitriyzaitsev
736 ● 9 ● 17

Why downvote? I had the same issue and fixed it exactly this way. – dmitriyzaitsev Oct 20, 2016 at 9:56

This works for me. BTW you need to use kapt: `compile 'org.parceler:parceler-api:1.1.6'` `kapt 'org.parceler:parceler:1.1.6'` – Nicolas Jafelle Nov 14, 2017 at 19:52

Specifying `size` with a default value (as mentioned above) should fulfill the need for the parameterless constructor, no? – rdoubleui Jan 3, 2018 at 14:52

4

if you use Kotlin 1.1.4 or above it's easier to use @Parcelize annotation

For doing this first add this to **build.gradle**

```
android {
    //other codes

    //for using latest experimental build of Android Extensions
    androidExtensions {
        experimental = true
    }
}
```

Then change your class like this

```
@Parcelize
data class Valve(val size: Int? = null) : Parcelable
```

Share

Improve this answer

Follow

edited Nov 28, 2019 at 9:33

answered May 12, 2019 at 11:40

Radesh

13.5k ● 4 ● 53 ● 65