Fragment Factory

E Técnicas de Injeção de Dependência

O que é?

Um Factory

De Fragments!

Como ele é

```
@NonNull
public Fragment instantiate(@NonNull ClassLoader classLoader, @NonNull String className) {
    try {
        Class<? extends Fragment> cls = loadFragmentClass(classLoader, className);
        return cls.getConstructor().newInstance();
    } catch (java.lang.InstantiationException e) {
        throw new Fragment.InstantiationException("Unable to instantiate fragment " + clas
                + ": make sure class name exists, is public, and has an"
                + " empty constructor that is public", e);
    } catch (IllegalAccessException e) {
```

Plugando na Activity

```
class MainActivity : AppCompatActivity() {

    [redacted]

    override fun onCreate(savedInstanceState: Bundle?) {
        [redacted]
        supportFragmentManager.fragmentFactory = ExampleFragmentFactory(
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

Construtor Vazio

```
fun getInstance(userId: Int): ExampleFragment {
    val bundle = Bundle()
    bundle.putInt("MINHA CHAVE", userId)
    val fragment = ExampleFragment()
    fragment.arguments = bundle
    return fragment
}
```

Construtor Vazio

```
fun getInstance(userId: Int, buttonClickCallback : () -> Unit): ExampleFragment {
    val bundle = Bundle()
    bundle.putInt("MINHA CHAVE", userId)
    val fragment = ExampleFragment()
    fragment.arguments = bundle
    fragment.setClickCallback(buttonClickCallback)
    return fragment
}
```

Quem usa

```
FragmentStateManager(@NonNull FragmentLifecycleCallbacksDispatcher dispatcher,
        @NonNull ClassLoader classLoader, @NotNull @NonNull FragmentFactory fra
        @NotNull @NonNull FragmentState fs) {
    mDispatcher = dispatcher;
    mFragment = fragmentFactory.instantiate(classLoader, fs.mClassName);
    if (fs.mArguments != null) {
        fs.mArguments.setClassLoader(classLoader);
    }
    mFragment.setArguments(fs.mArguments);
    mFragment.mWho = fs.mWho;
    mFragment.mFromLayout = fs.mFromLayout;
    mFragment.mRestored = true;
    mFragment.mHidden = fs.mHidden;
    mFragment.mMaxState = Lifecycle.State.values()[fs.mMaxLifecycleState];
    if (fs.mSavedFragmentState != null) {
        mFragment.mSavedFragmentState = fs.mSavedFragmentState;
    } else {
        // When restoring a Fragment, always ensure we have a
        // non-null Bundle so that developers have a signal for
```

FragmentState

```
final class FragmentState implements Parcelable {
    final String mClassName;
    final String mWho;
    final boolean mFromLayout;
    final int mFragmentId;
    final int mContainerId;
    final String mTag;
    final boolean mRetainInstance;
    final boolean mRemoving;
    final boolean mDetached;
    final Bundle mArguments;
    final boolean mHidden;
    final int mMaxLifecycleState;

    Bundle mSavedFragmentState;

    FragmentState( @NotNull Fragment frag) {
        mClassName = frag.getClass().getName();
        mWho = frag.mWho;
        mFromLayout = frag.mFromLayout;
        mFragmentId = frag.mFragmentId;
        mContainerId = frag.mContainerId;
```

mWho

```
// Internal unique name for this fragment;
@NonNull
String mWho = UUID.randomUUID().toString();
```

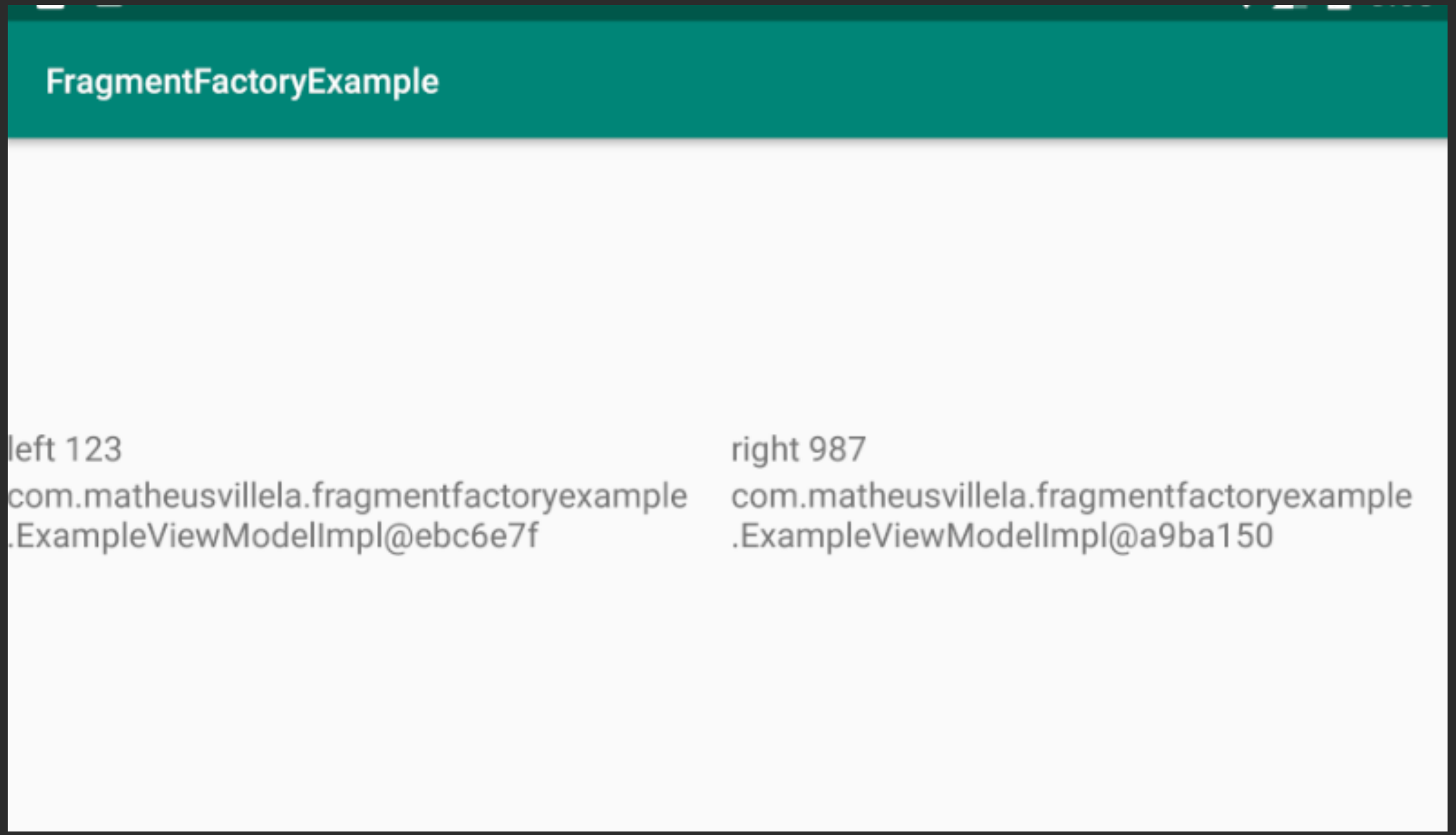
Ciclo Fragment/Activity

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        val transaction : FragmentTransaction = supportFragmentManager.beginTransaction()
        transaction.replace(
            R.id.activity_main_left_container,
            ExampleFragment::class.java, bundleOf( ...pairs: "argument" to ExampleArgument
        )
        transaction.replace(
            R.id.activity_main_right_container,
            ExampleFragment::class.java, bundleOf( ...pairs: "argument" to ExampleArgument
        )
        transaction.commit()
```

Cara da Activity



Ciclo Fragment/Activity

```kotlin
class ExampleFragment(

    init {
        Log.d( tag: "ExampleFragment", msg: "$this - init")
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        Log.d( tag: "ExampleFragment", msg: "$this - onViewCreated")
```

Ciclo Fragment/Activity

```
ExampleFragment{3fbce95} (bfd7d642-2d05-4c11-b893-d9b3e512ded1)} - init
ExampleFragment{15b604e} (29c5713f-4ecb-433b-b539-de8eac981d3f)} - init
ExampleFragment{3fbce95} (e0aac2a2-5ab0-4e7a-9d2e-5d59a8e11182) id=0x7f07003b} - onViewCreated
ExampleFragment{15b604e} (3f95676a-88d8-4e60-8ed5-ac5b571df2ee) id=0x7f07003c} - onViewCreated
```

```
ExampleFragment{15d5d7a} (effc6712-bdc9-47aa-80ba-1d9c5c73ac2c)} - init
ExampleFragment{33ed721} (ee3faf47-06aa-432c-b668-b1ee5ed635a7)} - init
ExampleFragment{92d53fc} (a6a4e84b-a30c-4361-97dd-3539bc506a54)} - init
ExampleFragment{75070a6} (c1ac5e29-3c77-4295-9033-e87cda89b218)} - init
ExampleFragment{15d5d7a} (e0aac2a2-5ab0-4e7a-9d2e-5d59a8e11182) id=0x7f07003b} - onViewCreated
ExampleFragment{33ed721} (3f95676a-88d8-4e60-8ed5-ac5b571df2ee) id=0x7f07003c} - onViewCreated
ExampleFragment{92d53fc} (4b8ef862-cbb1-409b-adfe-30f8e215fe2e) id=0x7f07003b} - onViewCreated
ExampleFragment{75070a6} (9ccf59b5-b9bf-403e-ae0f-0da07c357f81) id=0x7f07003c} - onViewCreated
```

Ciclo Fragment/Activity

```
class MainActivity : AppCompatActivity() {

 override fun onCreate(savedInstanceState: Bundle?) {


  if (savedInstanceState == null) {
      val transaction : FragmentTransaction = supportFragmentManager.beginTransaction()
      transaction.replace(
          R.id.activity_main_left_container,
          ExampleFragment::class.java, bundleOf( ...pairs: "argument" to ExampleArgument(
      )
      transaction.replace(
          R.id.activity_main_right_container,
          ExampleFragment::class.java, bundleOf( ...pairs: "argument" to ExampleArgument(
      )
      transaction.commit()
  }
```

Ciclo Fragment/Activity

```
ExampleFragment{3fbce95} (8bfe3cd5-97a0-4678-9e20-942bbde2c327)} - init
ExampleFragment{15b604e} (c57efc17-f9a8-4f5a-89fd-a8bd684fefa5)} - init
ExampleFragment{3fbce95} (a0100b27-5f29-4d4c-a874-8f046641d867) id=0x7f07003b} - onVie
ExampleFragment{15b604e} (0f921988-7743-4ad4-a7e3-d3c678d1f61e) id=0x7f07003c} - onVie
ExampleFragment{15d5d7a} (58afeddf-e333-42e4-a167-6b39fd4239e5)} - init
ExampleFragment{33ed721} (b6a73520-da11-42cb-ab5b-1260cfd9ecd1)} - init
ExampleFragment{33ed721} (a0100b27-5f29-4d4c-a874-8f046641d867) id=0x7f07003b} - onVie
ExampleFragment{15d5d7a} (0f921988-7743-4ad4-a7e3-d3c678d1f61e) id=0x7f07003c} - onVie
```

Voltando um pouco...

```
@NonNull
public Fragment instantiate(@NonNull ClassLoader classLoader, @NonNull String className) {
    try {
        Class<? extends Fragment> cls = loadFragmentClass(classLoader, className);
        return cls.getConstructor().newInstance();
    } catch (java.lang.InstantiationException e) {
        throw new Fragment.InstantiationException("Unable to instantiate fragment " + clas
                + ": make sure class name exists, is public, and has an"
                + " empty constructor that is public", e);
    } catch (IllegalAccessException e) {
```

Isso explica muita coisa

136898243 Can't inject a parameter specific to a fragment using FragmentFactory [AOSP] assigned

3 people have starred this issue.

Matheus Villela <matheusvillela@gmail.com> #2

This is a deal breaker issue that I'm currently facing, there's just no way to correctly scope Fragment's dependencies.

il...@google.com <il...@google.com> #24

You should only be injecting stateless objects via FragmentFactory such as a repository object, OkHttpClient, etc

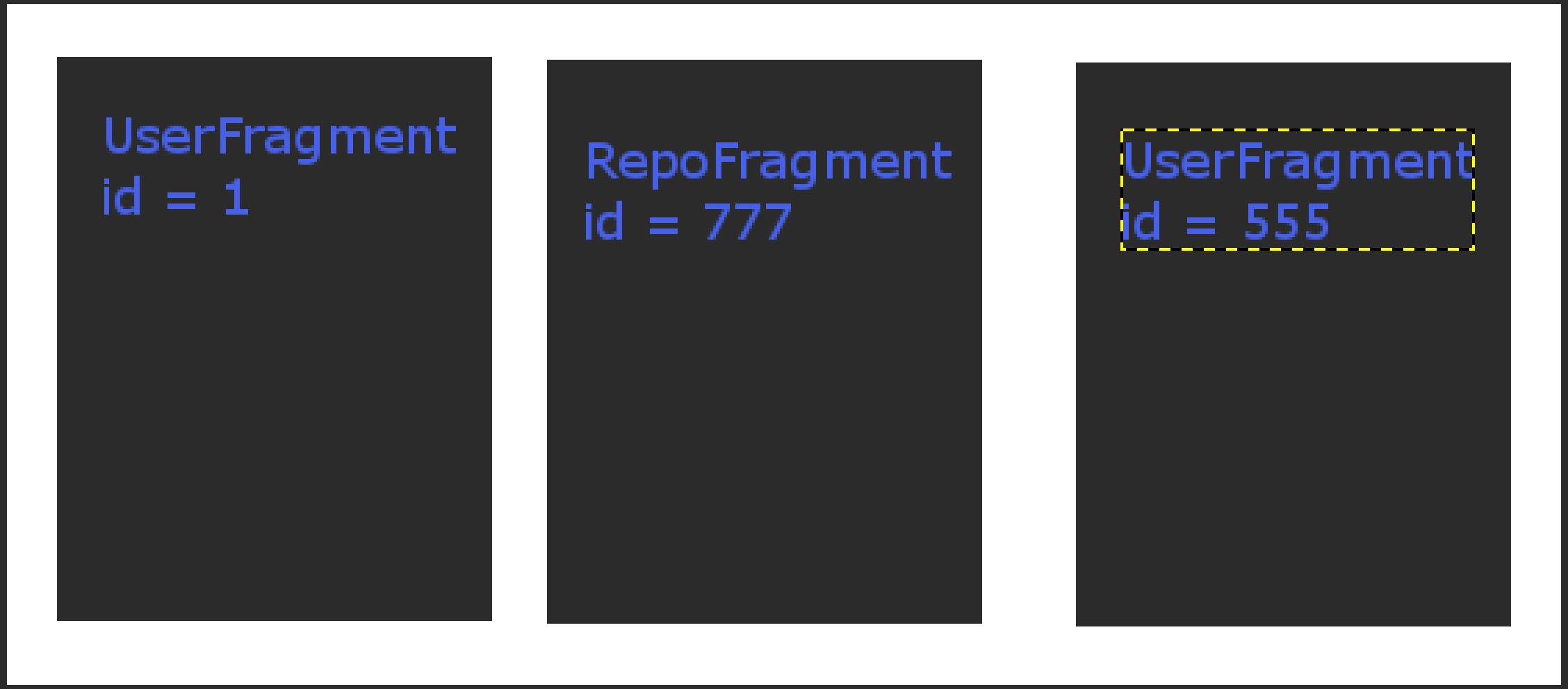
Voltando mais...

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        val transaction : FragmentTransaction = supportFragmentManager.beginTransaction()
        transaction.replace(
            R.id.activity_main_left_container,
            ExampleFragment::class.java, bundleOf( ...pairs: "argument" to ExampleArgument
        )
        transaction.replace(
            R.id.activity_main_right_container,
            ExampleFragment::class.java, bundleOf( ...pairs: "argument" to ExampleArgument
        )
        transaction.commit()
```

Empilhamento



Formas de Empilhamento

Add/remove (show/hide)

Attach/Detach

Replace

Google sendo Google

Posted by u/Zhuinden EpicPandaForce @ SO 2 months ago

189

It's confirmed that Fragment/FragmentManager functionality will be pruned to only support "add", "remove", and "replace", because that is all that Jetpack Navigation needs (and no other use-case will be supported)

News

After having a chat with Ian Lake, apparently the only way to keep a Fragment alive along with its ViewModelStore will be to have ~~the Fragment~~ the FragmentTransaction that keeps the Fragment alive on the FragmentManager's backstack: https://twitter.com/ianhlake/status/1189166861230862336

This also brings forth the following deprecations:

- `Fragment.setRetainInstance`
- `FragmentTransaction.attach`/`FragmentTransaction.detach`
- `FragmentTransaction.show`/`FragmentTransaction.hide`
- `FragmentPagerAdapter`

At this point, one might wonder why they didn't just create a new UI component.

145 Comments Give Award Share Save Hide Report 99% Upvoted

Meta

```
@InjectConstructor
class ExampleFragment(
    private val viewModel: ExampleViewModel,
    private val navigator : MainActivity2.Navigator
) : Fragment() {
```

```
data class ExampleArgument(val identifier : String) : java.io.Serializable
```

```
@InjectConstructor
class ExampleViewModelImpl(argument: ExampleArgument) : ExampleViewModel {
    override val identifier: BehaviorSubject<String> =
        BehaviorSubject.createDefault(argument.identifier)
    override val objectString: BehaviorSubject<String> =
        BehaviorSubject.createDefault(this.toString())
}
```

Dores

```
// Just get it
class MyActivity() : AppCompatActivity() {

  // lazy inject MyViewModel
  val myViewModel : MyViewModel by viewModel()
}
```

```
private val viewModel2 = lazy {
    ServiceLocator.getInstance().get<ExampleViewModel>(this)
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val userId : Int = arguments?.getInt( key: "extras-user-id") ?: throw Ille
    viewModel.setUserId(userId)
}
```

Ajustando Activity

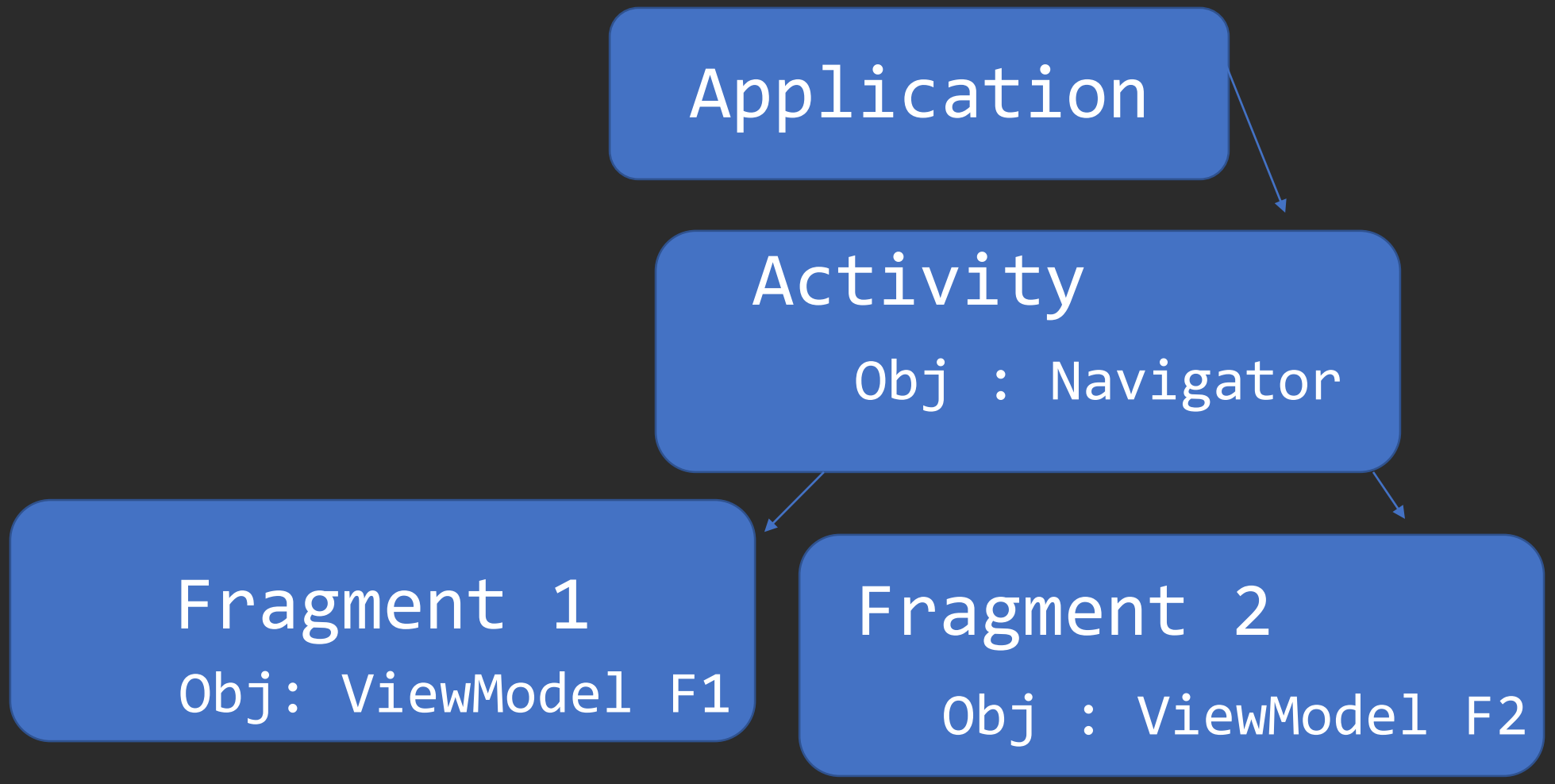
```
class MainActivity2 : AppCompatActivity() {

    private lateinit var scopeUuid: String

    override fun onCreate(savedInstanceState: Bundle?) {
        scopeUuid = savedInstanceState?.getString( key: "scope_uuid") ?: UUID.randomUUID().toString()
        val subScope : Scope! = KTP.openRootScope()
            .openSubScope(scopeUuid)
        supportFragmentManager.fragmentFactory = ExampleFragmentFactory(subScope)

    override fun onSaveInstanceState(outState: Bundle) {
        outState.putString("scope_uuid", scopeUuid)
        super.onSaveInstanceState(outState)
    }
```

DI por Escopo



Ajustando Androidx

```
FragmentStateManager(@NonNull FragmentLifecycleCallbacksDispatcher dispatcher,
        @NonNull ClassLoader classLoader, @NotNull @NonNull FragmentFactory fragmentFactory,
        @NotNull @NonNull FragmentState fs) {
    mDispatcher = dispatcher;
    mFragment = fragmentFactory.instantiateWithArguments(classLoader, fs.mClassName,
            fs.mWho, fs.mArguments);
```

```
public class FragmentFactory {

    @NonNull
    public Fragment instantiateWithArguments(@NonNull ClassLoader classLoader,
            @NonNull String className,
            @Nullable @NonNull String mWho, @Nullable @Nullable Bundle arguments) {
        return instantiate(classLoader, className);
    }
```

Nosso Fragment Factory!

```
class ExampleFragmentFactory(private val scope: Scope) : FragmentFactory() {
    override fun instantiate(classLoader: ClassLoader, className: String): Fragment {
        return createFragment(classLoader, className, mWho: null, arguments: null)
    }

    override fun instantiateWithArguments(
        classLoader: ClassLoader, className: String,
        mWho: String, arguments: Bundle?
    ): Fragment {
        return createFragment(classLoader, className, mWho, arguments)
    }
```

Nosso Fragment Factory!

```
private fun createFragment(
    classLoader: ClassLoader, className: String,
    mWho: String?, arguments: Bundle?
): Fragment {
    val cls : Class<out Fragment!> = loadFragmentClass(classLoader, className)
    val who : String = mWho ?: UUID.randomUUID().toString()
    val subScope : Scope! = scope.openSubScope(who) { it: Scope!
        it.installModules(object : Module() {
            init {
                arguments?.let { it: Bundle
                    val obj: Serializable? = it.getSerializable( key: "argument")
                    if (obj != null) {
                        bind(obj.javaClass).toInstance(obj)
                    }
                }
            }
        }, ViewModelModule())
    }
```

DI não precisa ser difícil

```kotlin
class ViewModelModule : Module() {

    init {
        bind(ExampleViewModel::class.java)
            .to(ExampleViewModelImpl::class.java).singleton()
    }
}
```

```kotlin
interface ExampleViewModel {
    val identifier : Observable<String>
    val objectString : Observable<String>
}
```

```kotlin
@InjectConstructor
class ExampleViewModelImpl(argument: ExampleArgument) : ExampleViewModel {
    override val identifier: BehaviorSubject<String> =
        BehaviorSubject.createDefault(argument.identifier)
    override val objectString: BehaviorSubject<String> =
        BehaviorSubject.createDefault(this.toString())
}
```

Bundle Argument

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        val transaction : FragmentTransaction = supportFragmentManager.beginTransaction()
        transaction.replace(
            R.id.activity_main_left_container,
            ExampleFragment::class.java, bundleOf( ...pairs: "argument" to ExampleArgument(
        )
        transaction.replace(
            R.id.activity_main_right_container,
            ExampleFragment::class.java, bundleOf( ...pairs: "argument" to ExampleArgument(
        )
        transaction.commit()
```

Nosso Fragment Factory!

```
private fun createFragment(
    classLoader: ClassLoader, className: String,
    mWho: String?, arguments: Bundle?
): Fragment {
    val cls : Class<out Fragment!> = loadFragmentClass(classLoader, className)
    val who : String = mWho ?: UUID.randomUUID().toString()
    val subScope : Scope! = scope.openSubScope(who) { it: Scope!
        it.installModules(object : Module() {
            init {
                arguments?.let { it: Bundle
                    val obj: Serializable? = it.getSerializable( key: "argument")
                    if (obj != null) {
                        bind(obj.javaClass).toInstance(obj)
                    }
                }
            }
        }, ViewModelModule())
    }
```

Nosso Fragment Factory!

```
        }, ViewModelModule())
    }
    val fragment : Fragment! = subScope.getInstance(cls)
    fragment.lifecycle.addObserver(object : LifecycleObserver {
        @OnLifecycleEvent(Lifecycle.Event.ON_DESTROY)
        fun onDestroy() {
            if (fragment.isRemoving) {
                KTP.closeScope(subScope)
                fragment.lifecycle.removeObserver( observer: this)
            }
        }
    })
    return fragment
```

onCleared

```
val publisher : OnClearedPublisher! = subScope.getInstance(OnClearedPublisher::cl
val fragment : Fragment! = subScope.getInstance(cls)
fragment.lifecycle.addObserver(object : LifecycleObserver {
    @OnLifecycleEvent(Lifecycle.Event.ON_DESTROY)
    fun onDestroy() {
        if (fragment.isRemoving) {
            KTP.closeScope(subScope)
            fragment.lifecycle.removeObserver( observer: this)
            publisher.publish()
        }
    }
})
```

onCleared

```
lass ViewModelModule : Module() {

    init {
        bind(ExampleViewModel::class.java)
            .to(ExampleViewModelImpl::class.java).singleton()
        bind(OnClearedSubscriber::class.java).singleton()
        bind(OnClearedPublisher::class.java).singleton()
    }
```

onCleared

```
@InjectConstructor
class ExampleViewModelImpl(
    argument: ExampleArgument,
    onClearedSubscriber: OnClearedSubscriber) : ExampleViewModel {
    override val identifier: BehaviorSubject<String> =
        BehaviorSubject.createDefault(argument.identifier)
    override val objectString: BehaviorSubject<String> =
        BehaviorSubject.createDefault(this.toString())

    init {
        onClearedSubscriber.subscribe(object : OnClearedSubscriber.Subscriber {
            override fun onCleared() {
                // bla
            }
        })
    }
}
```

Fim