Rethinking Dependency Injection
Producer-Consumer Linking at the Bytecode Level

Junyu Yue

Mobile Infrastructure @ TikTok

TikTok

Agenda

1. Challenges of Hilt
2. TikTok DI Solution
3. Open Source & Future Plans

TikTok

Challenges of Hilt in TikTok

Build Time Impact

10% of TikTok compilation duration

TikTok

Multiple Rounds Annotation Processing

TikTok

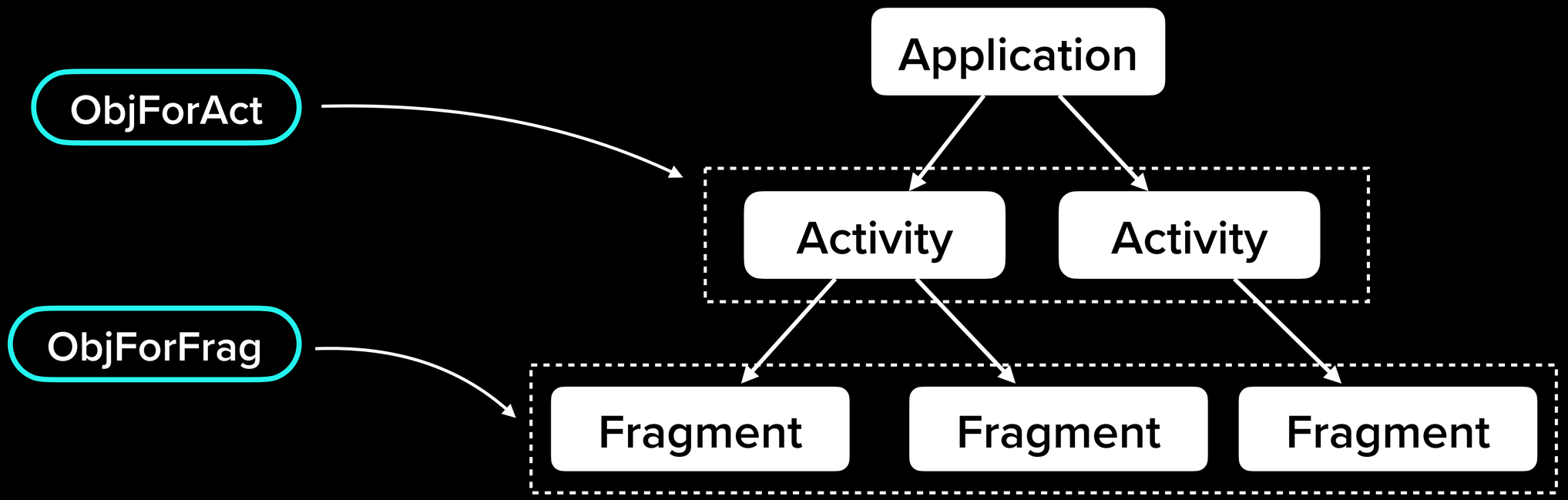
Multiple Rounds Annotation Processing

TikTok

Multiple Rounds Annotation Processing

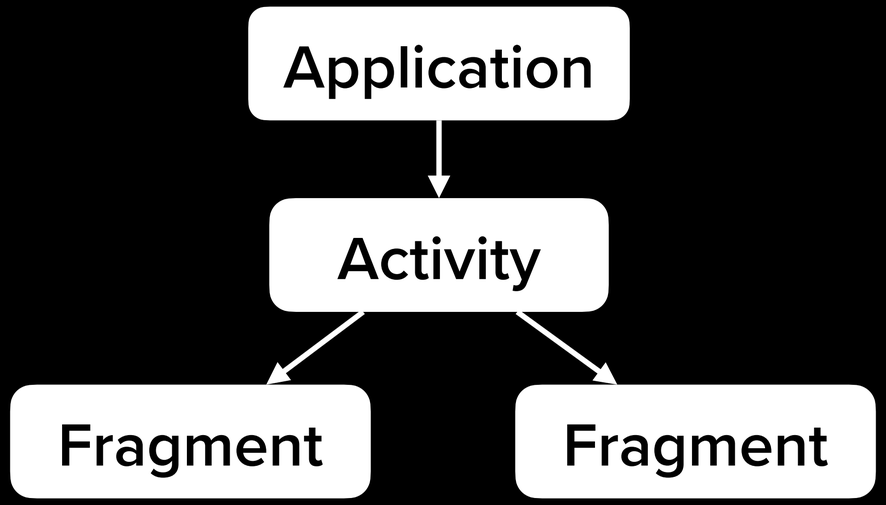
TikTok

Hilt Standardized Hierarchy



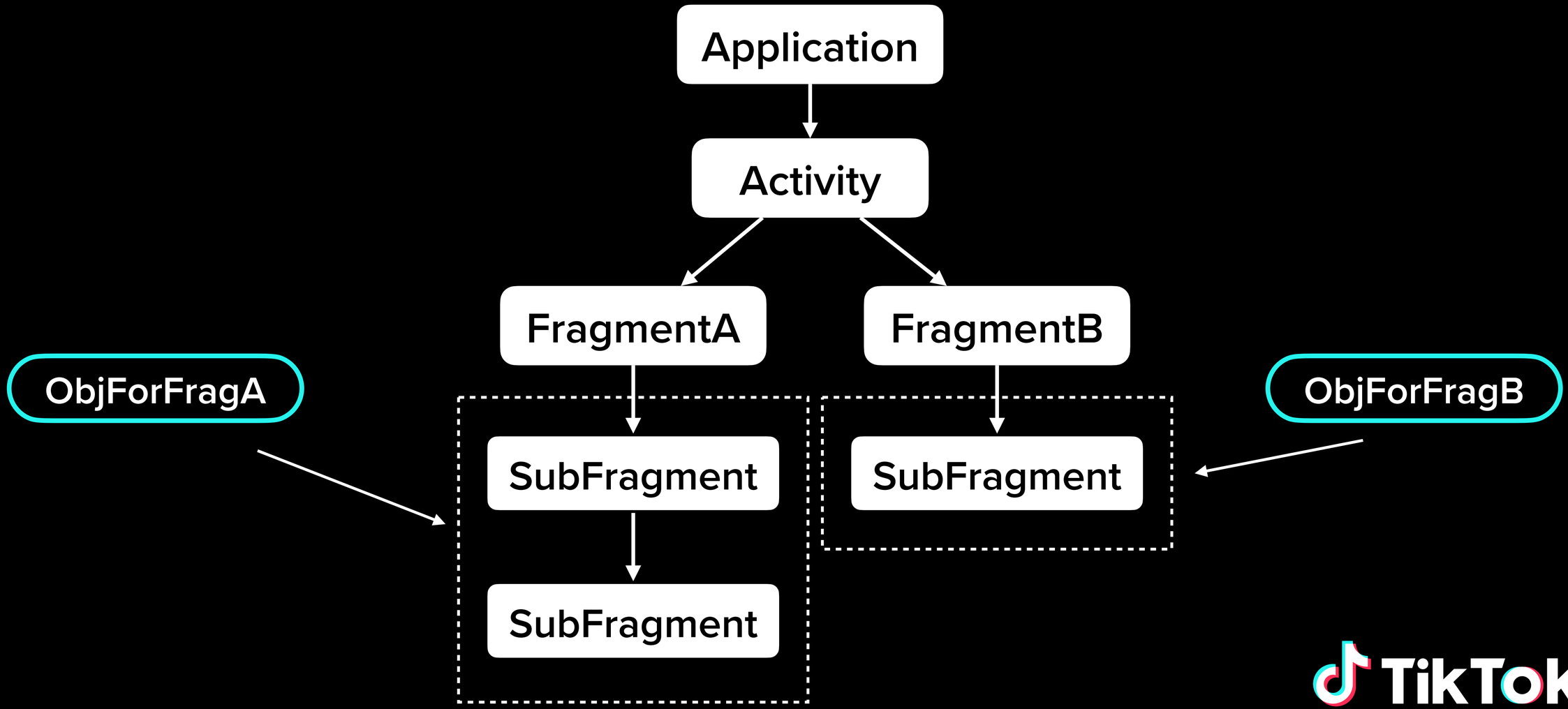
TikTok

TikTok Hierarchy



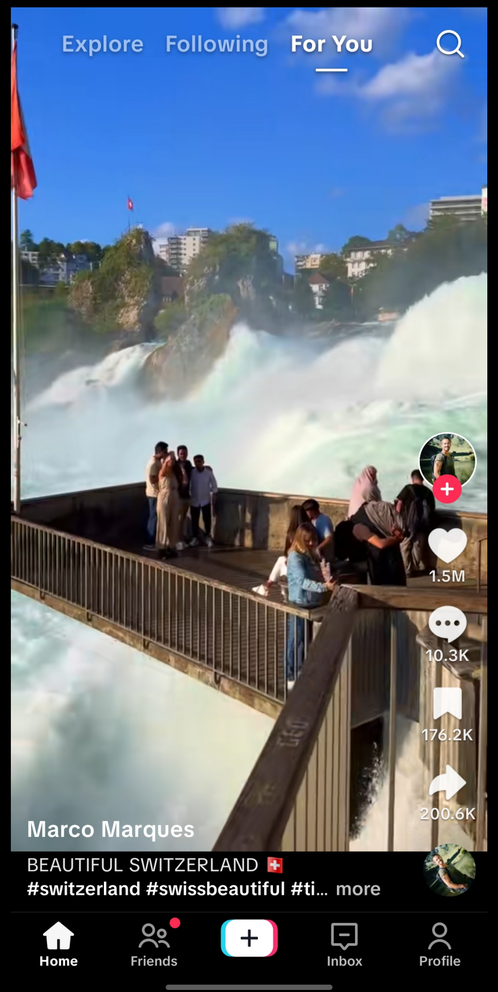
TikTok

TikTok Hierarchy

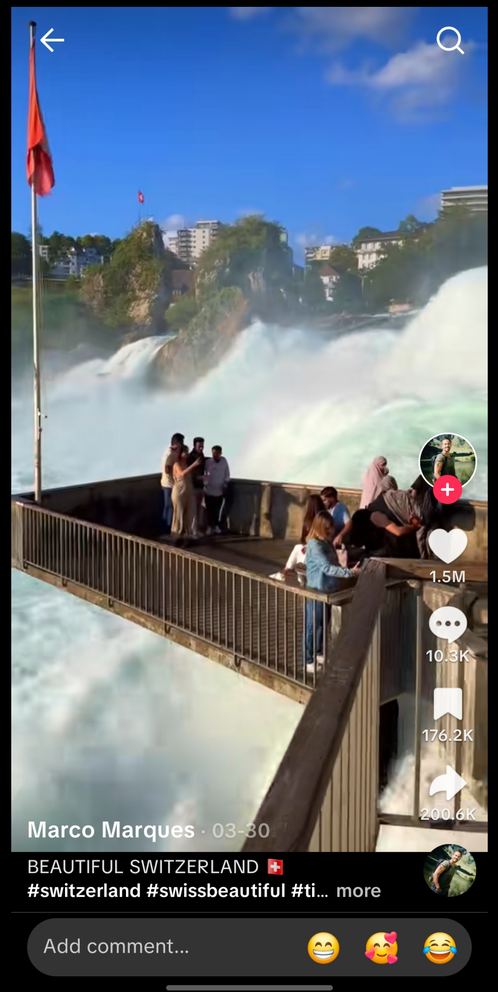


TikTok

FeedFragment



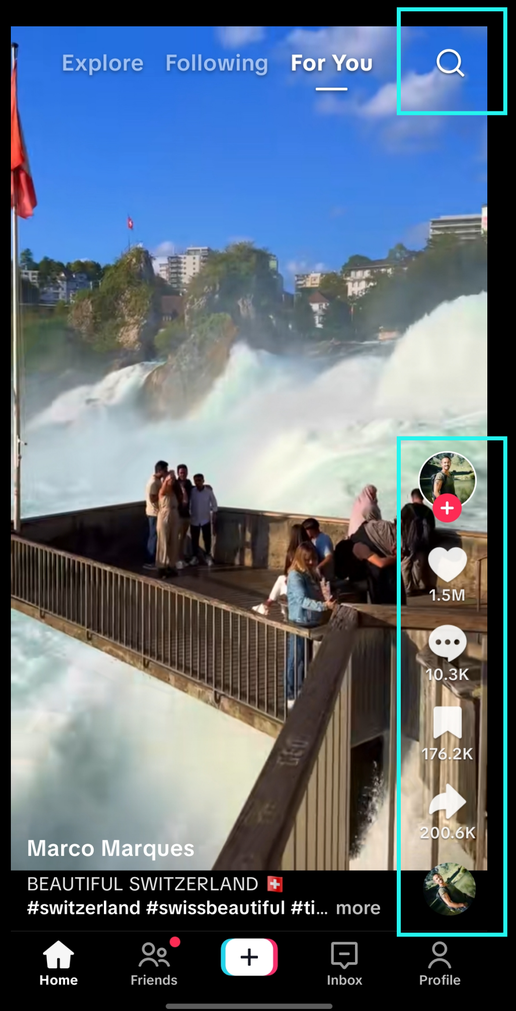
DetailFragment

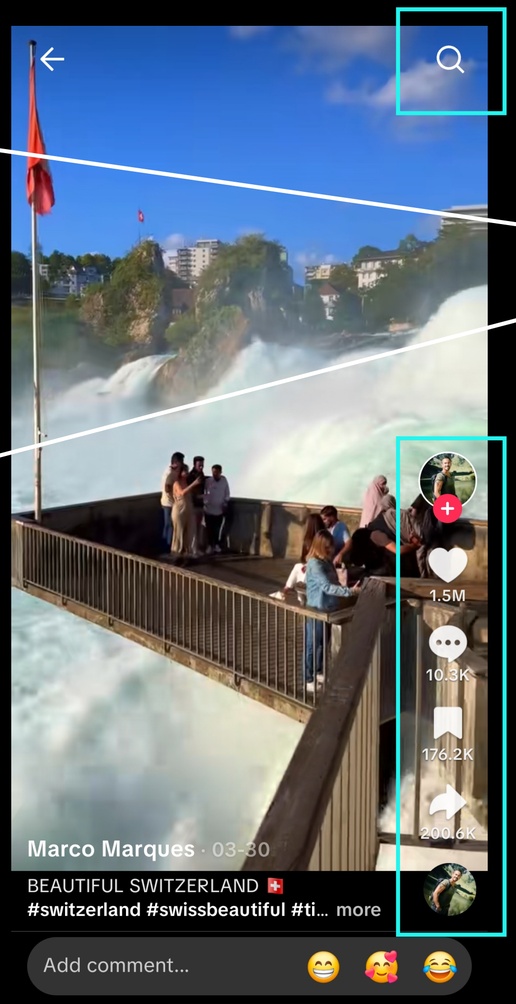


TikTok

FeedFragment

DetailFragment





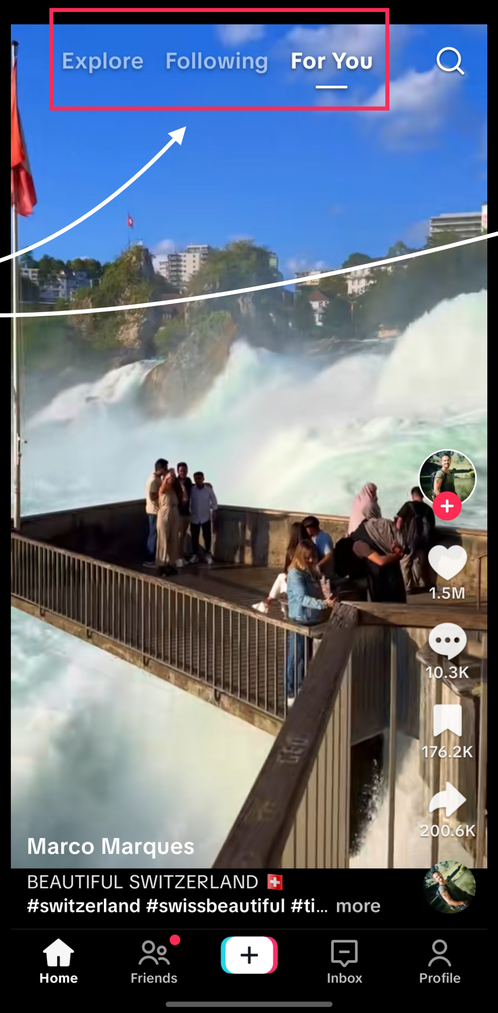
Same

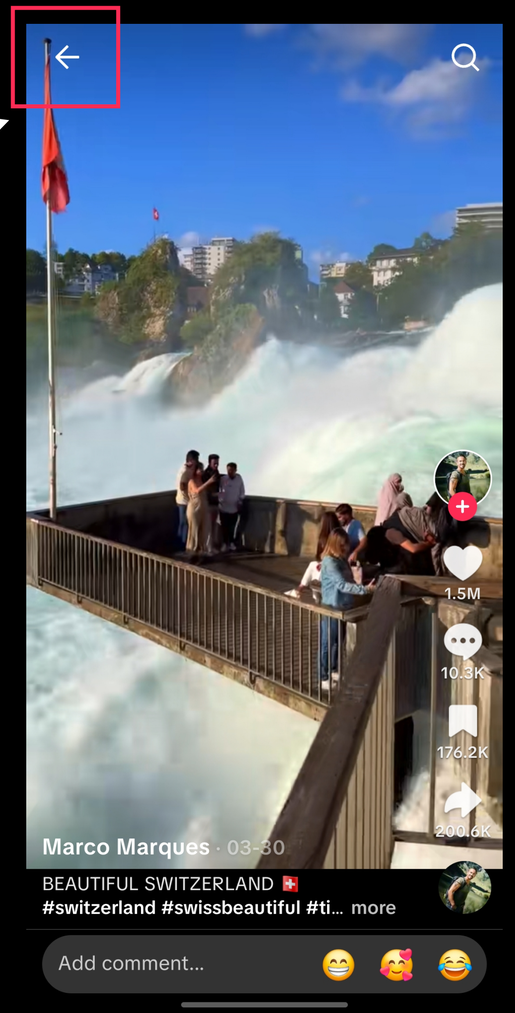
TikTok

FeedFragment

DetailFragment

Different





TikTok

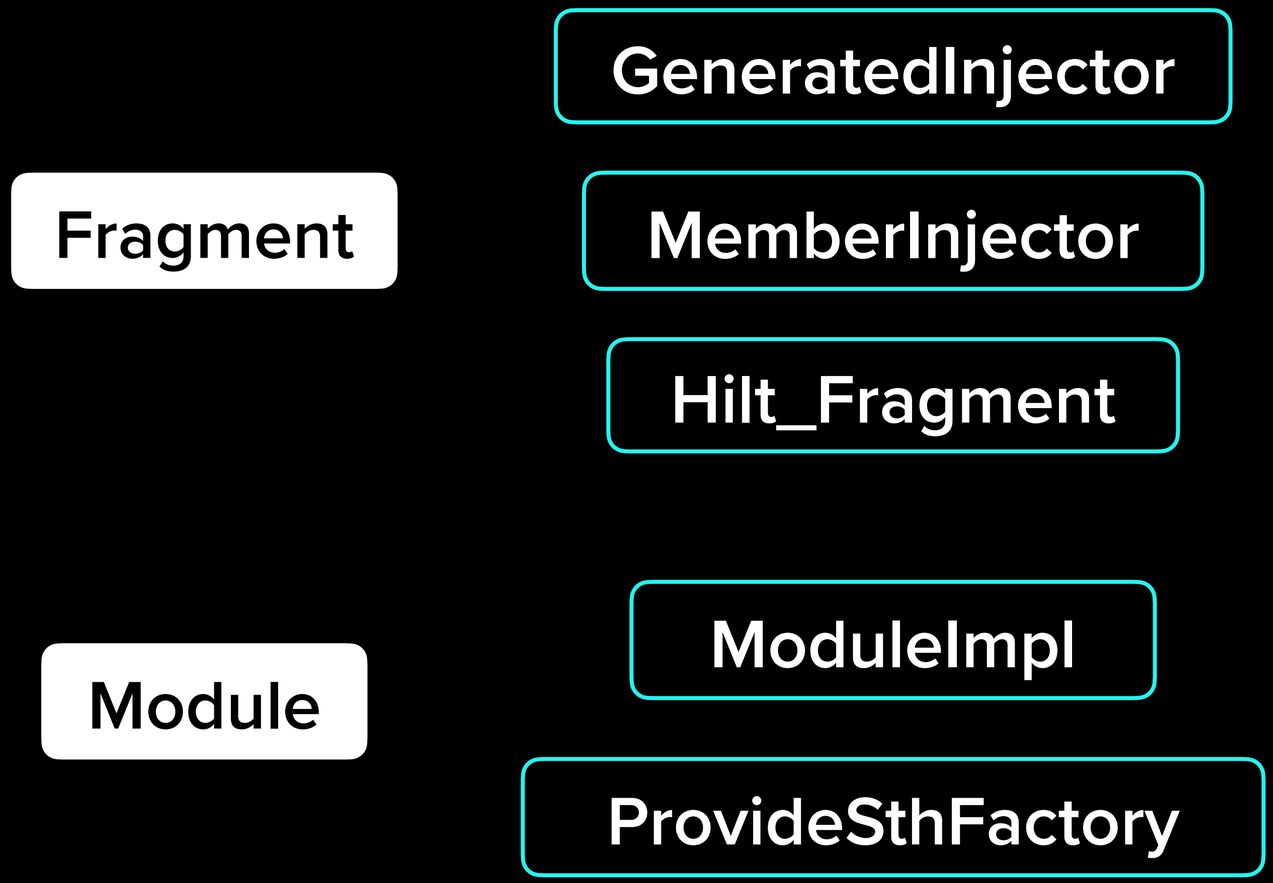
Runtime Loading Overhead

388K ANR per month

Top 3 in TikTok

TikTok

Package Size



TikTok

What’s the Ideal DI Pattern?

Fast

Fast compile-time & Fast runtime

Flexible

Flexible component hierarchy

Safe

Verify dependency errors
at compile stage

Minimized

Minimal code generated

TikTok

What is the basis of DI?

What is the basis of DI?

Producer

Consumer

TikTok

What is the basis of DI?

TikTok

TikTok DI Solution

 Knit

Knit

```
@Provides
class UserRepository

class UserService {
    val repository: UserRepository by di
}
```

TikTok

Knit

```
@Provides
class UserRepository

class UserService {
    val repository: UserRepository by di
}
```

TikTok

Knit

```
@Provides
class UserRepository

class UserService {
    val repository: UserRepository by di
}
```

TikTok

Knit

```
@Provides
class UserRepository

class UserService {
    val repository: UserRepository by di
}
```

TikTok

Knit

```
@Provides
class UserRepository

class UserService {
   val repository: UserRepository by di
}
```

TikTok

Knit

```
@Provides
class UserRepository

class UserService {
   val repository: UserRepository by di
}
```

After Knit Transform

```
class UserRepository

class UserService {
   val repository: UserRepository get() {
      return UserRepository()
   }
}
```

TikTok

Knit

```
@Provides
class UserRepository

class UserService {
   val repository: UserRepository by di
}
```

After Knit Transform

```
class UserRepository

class UserService {
  val repository: UserRepository get() {
    var localRepo = this._repository
    if (localRepo != null) return localRepo
    synchronized(this) {
      localRepo = this._repository
      if (localRepo != null) return localRepo
      localRepo = UserRepository()
      _repository = localRepo
      return localRepo
    }
  }
}
```

TikTok

Knit

```
class UserRepository

class UserService {
  val repository: UserRepository get() {
    var localRepo = this._repository
    if (localRepo != null) return localRepo
    synchronized(this) {
      localRepo = this._repository
      if (localRepo != null) return localRepo
      localRepo = UserRepository()
      _repository = localRepo
      return localRepo
    }
  }
}
```

```
.method public final
getRepository()Lknit/demo/UserRepository;
   aload 0 # "this"
   getfield knit/demo/UserService _repository
   astore 1 # val localRepo = this._repository
   aload 1
   ifnonnull :L4

   aload 0
   monitorenter # synchronized(this)
 :L0
   aload 0
   getfield knit/demo/UserService _repository
   astore 1 # localRepo = this._repository
   aload 1
   ifnonnull :L3
 :L1
   anew knit/demo/UserRepository
   dup
   invokespecial knit/demo/UserRepository <init> ()V
   astore 1 # localRepo = UserRepository()
 :L2
   aload 0
   aload 1
   putfield knit/demo/UserService _repository
 :L3
   aload 0
   monitorexit
 :L4
   aload 1
   areturn
.end method
```

TikTok

Knit

```
@Provides
class UserRepository

class UserService {
    val repository: UserRepository by di
}
```

TikTok

@Provides with Arguments

```
@Provides
class UserRepository(val userName: String)

class UserService {

    @Provides
    private fun provideName(): String = “bob”

    val repository: UserRepository by di
}
```

TikTok

@Provides with Arguments

```
@Provides
class UserRepository(val userName: String)

class UserService {

    @Provides
    private fun provideName(): String = “bob”

    val repository: UserRepository by di
                    -> UserRepository(provideName())
}
```

TikTok

Provided by Parent Component

Provided by Parent Component

```
class MainActivityComponent(
    @Provides val userRepository: UserRepository
)

class ProfileComponent(
    @Component val main: MainActivityComponent
) {
    val userRepository: UserRepository by di
}
```

TikTok

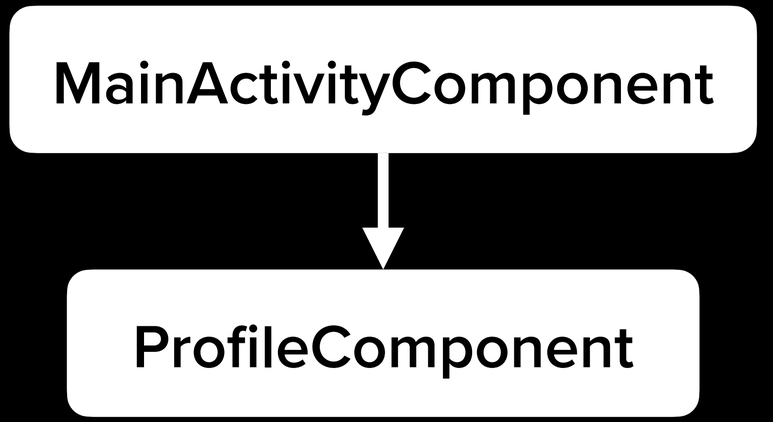
Provided by Parent Component

```
class MainActivityComponent(
    @Provides val userRepository: UserRepository
)

class ProfileComponent(
    @Component val main: MainActivityComponent
) {
    val userRepository: UserRepository by di
                        -> main.userRepository
}
```

TikTok

Provided by Parent Component

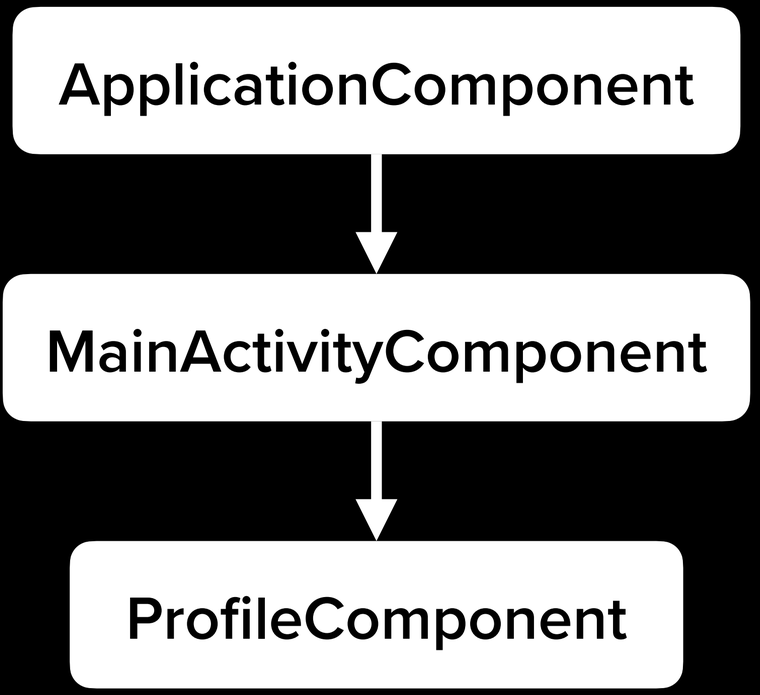
```
class MainActivityComponent(
    @Provides val userRepository: UserRepository
)

class ProfileComponent(
    @Component val main: MainActivityComponent
) {
    val userRepository: UserRepository by di
                        -> main.userRepository
}
```



TikTok

🧬 Provided by Parent Component

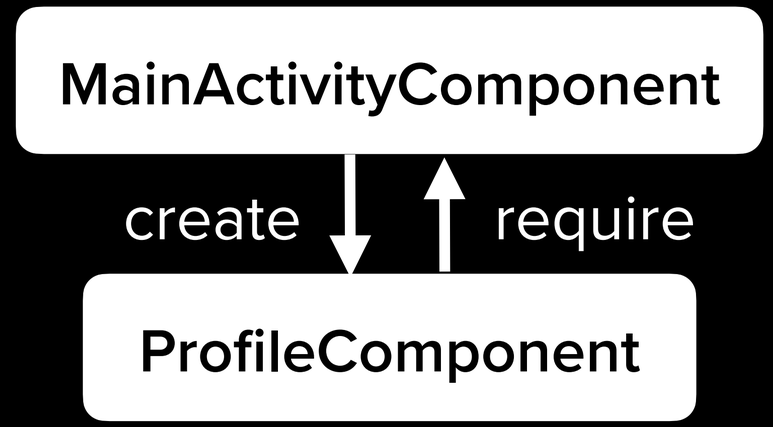
```
class ApplicationComponent(
    @Provides val application: Application,
)

class MainActivityComponent(
    @Component val appComponent: ApplicationComponent,
    @Provides val userRepository: UserRepository,
)

class ProfileComponent(
    @Component val main: MainActivityComponent,
) {
    val application: Application by di
    val userRepository: UserRepository by di
}
```



TikTok

🧬 Provided by Parent Component

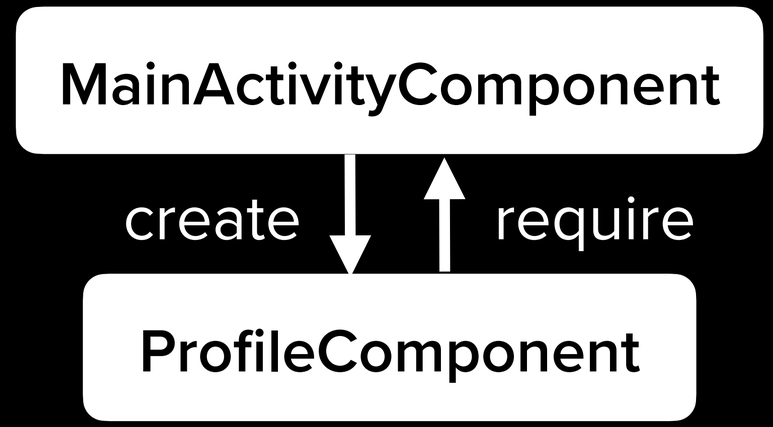
```
class MainActivityComponent(
    @Provides val userRepository: UserRepository
) {
    val profile = ProfileComponent(this)
}

class ProfileComponent(
    @Component val main: MainActivityComponent
) {
    val userRepository: UserRepository by di
}
```



TikTok

Provided by Parent Component

```
// main module
class MainActivityComponent(
    @Provides val userRepository: UserRepository
) {
    val profile = ProfileComponent(this)
}

// profile module
class ProfileComponent(
    @Component val main: MainActivityComponent
) {
    val userRepository: UserRepository by di
}
```

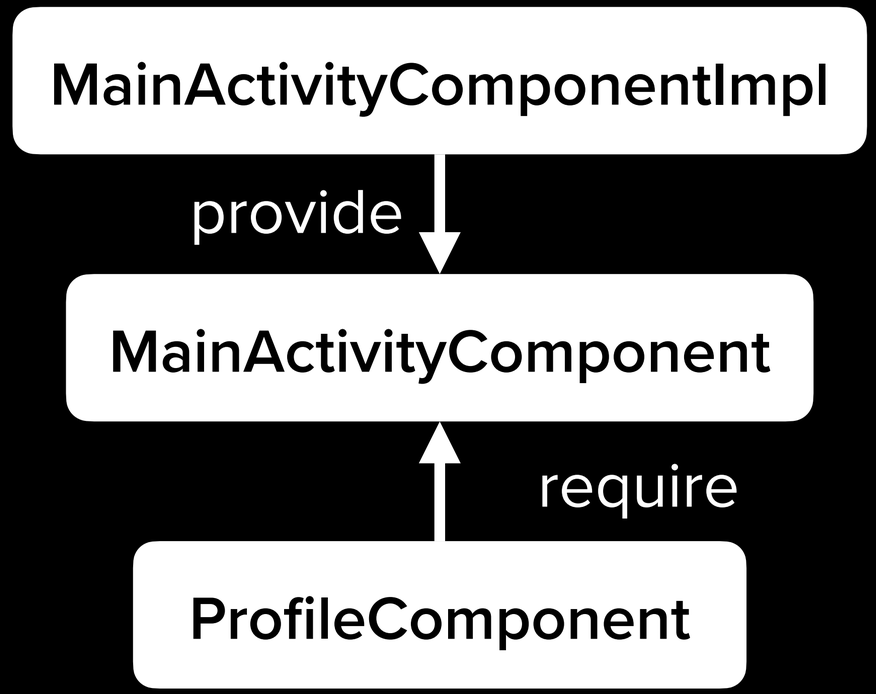


TikTok

Interface Injection

Interface Injection

```
// main-api module
interface MainActivityComponent {
    @get:Provides
    val userRepository: UserRepository
}

// main-impl module
@Provides(MainActivityComponent::class)
class MainActivityComponentImpl(
    override val userRepository: UserRepository
) : MainActivityComponent

// profile module
class ProfileComponent(
    @Component val main: MainActivityComponent
) {
    val userRepository: UserRepository by di
}
```



TikTok

More Features

More Features

- Multi-bindings
- Singleton
- ViewModel Injection
- Factory / Lazy / Loadable

TikTok

🧬 More Features

- Multi-bindings
- Singleton
- ViewModel Injection
- Factory / Lazy / Loadable

```
@Provides(Repo::class)
class RemoteRepo: Repo

@Provides(Repo::class)
class LocalRepo: Repo

val repos: List<Repo> by di
```

TikTok

More Features

- Multi-bindings
- Singleton
- ViewModel Injection
- Factory / Lazy / Loadable

```
var i: Int = 1

@Provides
@Singleton
fun provideInt(): Int = i++

val a: Int by di // 1
val b: Int by di // 1
```

TikTok

🧬 More Features

- Multi-bindings
- Singleton
- ViewModel Injection
- Factory / Lazy / Loadable

```
class FooViewModel
    @KnitViewModel constructor()
: ViewModel()

@Component
class MyFragment : Fragment() {
    val fooVM: FooViewModel
        by knitViewModel()
}
```

TikTok

More Features

- Multi-bindings
- Singleton
- ViewModel Injection
- Factory / Lazy / Loadable

```
@Provides
class Foo

val factory: Factory<Foo> by di
val lazy: Lazy<Foo> by di
val loadable: Loadable<Foo> by di
```

TikTok

More Features

- Multi-bindings
- Singleton
- ViewModel Injection
- Factory / Lazy / Loadable

```
@Provides
class Foo

val factory: () -> Foo by di
val lazy: kotlin.Lazy<Foo> by di
val loadable: Loadable<Foo> by di
```

TikTok

Benefits After Using Knit

Benefits After Using Knit

Build time reduced by 10%

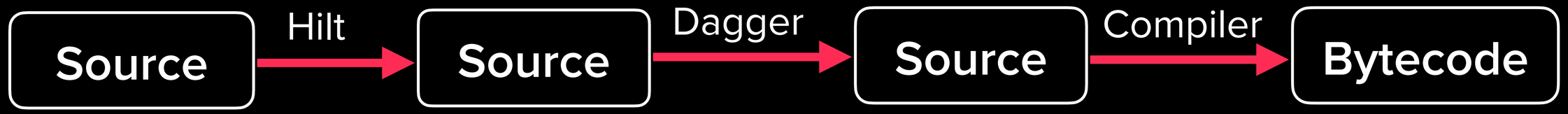
ANR reduced 388K per month

Package size reduced 1M

TikTok

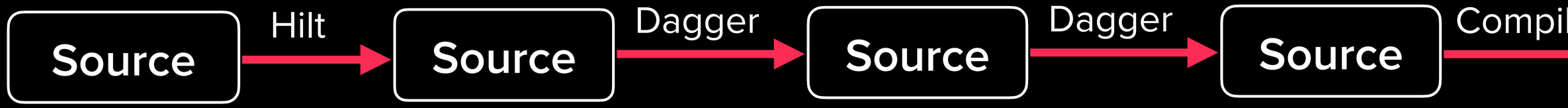
Compare with Hilt

Compare with Hilt



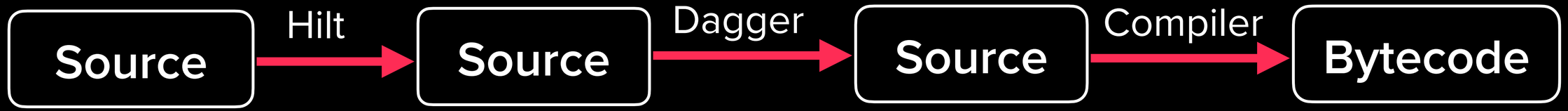
TikTok

Compare with Hilt



TikTok

Compare with Hilt



TikTok

Smaller Package

Smaller Package

Fragment

Hilt

Module

TikTok

Smaller Package

Hilt

Fragment

Module

GeneratedInjector

MemberInjector

Hilt_Fragment

ModuleImpl

ProvideSthFactory

TikTok

Smaller Package

Fragment

Knit

Component

TikTok

Smaller Package

Knit

Fragment

TikTok

Runtime Performance

Runtime Performance

Everything Lazily by Default

Limitations

Limitations

- Kotlin only
- Not compatible with JSR330 @Inject
- No sources generated

TikTok

Future Plans

Future Plans

- Inject functions

TikTok

Future Plans

- Inject functions

```
@Provides
fun provideFooByKey(key: String): Foo = Foo(key)

class Container {
    val fooFactory: (String) -> Foo by di
}
```

TikTok

Future Plans

- Kotlin Multiplatform support

TikTok

Future Plans

- Inlay hint in IDE

```
class Parent {
    @Provides
    val bar: Bar = ...
}

class Foo {
    @Component
    val parent: Parent = Parent()

    val bar: Bar by di = parent.bar
}
```

TikTok

Open Source

Open Source

github.com/tiktok/knit