Dependency injection made easy with Dagger2

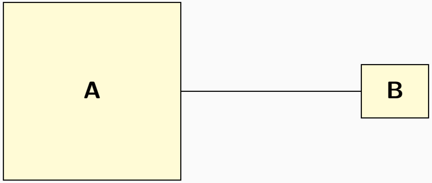
Alessandro Candolini

January 25, 2018

Agenda

Dependency injection principles

What is a dependency?



```
/** Class A */
class A {
    // ....
    fun doSomething() {
        b.log("text")
    }
}

/** Class B (dependency) */
class B {
    fun log(text : String) {
    }
}
```

```kotlin
// Option 1 - static methods

class A {
    fun doSomething() {
        B.log("text") // <- static method
    }
}

class B {
    companion object {
        fun log(text: String) {
        }
    }
}
```

Examples:

- Helper classes
- Utils classes
- Manager classes, etc...

Drawbacks:

- *A* not testable in isolation
- *A tightly coupled* to *B*
- Lack of encapsulation (backdoor)
- *Hidden* dependency

More Examples:

- `Application.getStaticContext()\`
- migrating one class leads to migrate 100 classes. . .

```kotlin
// Option 2 - singletons

class A {
    fun doSomething() {
        B.log("text") // <- singleton
    }
}

object B {
    fun log(text: String) {
    }
}
```

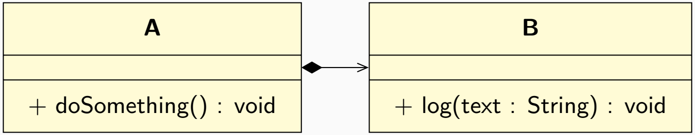
```
// Option 3 - composition

class A {
    private val b : B = B() // <-- instantiate

    fun doSomething() {
        b.log("text")
    }
}

class B {
    fun log(text: String) {
    }
}
```

Composition



Dagger2

Dagger2 Android

Questions?