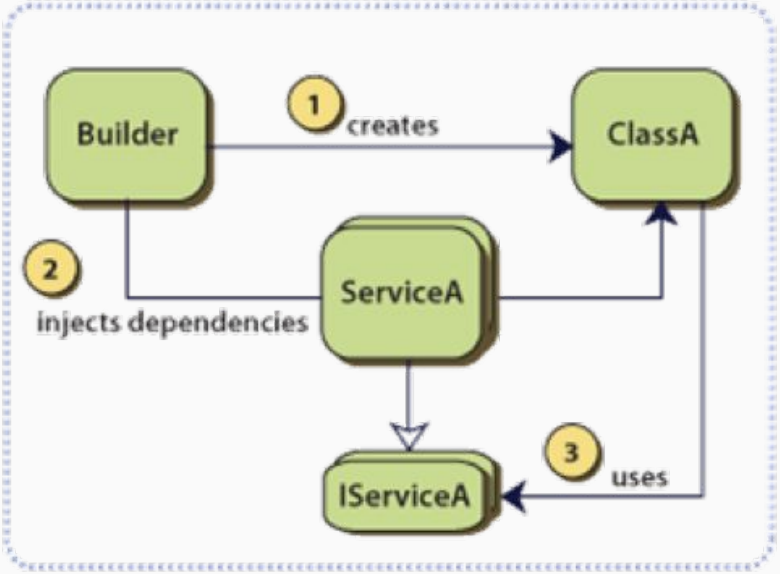
Dependency Injection with Koin

Anuj Middha
@anujmiddha

Dependency Injection



Simplest Dependency Injection

Dependency Injection Frameworks on Android

- Dagger
- Guice
- Toothpick
- Koin
- Kodein

INSERT KOIN®

Koin

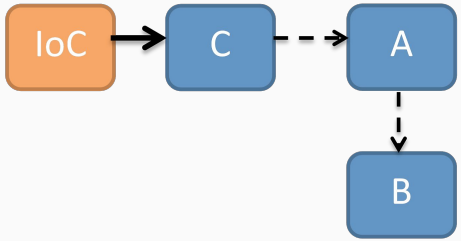
Koin is pragmatic lightweight dependency injection framework for Kotlin developers. Written in pure Kotlin, using functional resolution only: no proxy, no code generation, no reflection. Koin is a DSL, a lightweight container and a pragmatic API.

Dependency Injection vs Service Locator

Class Dependencies

Service Location / Active Calling

IoC / DI / Auto-Wiring / Passive Calling



Koin DSL

- module { } - Create a Koin module or a submodule
- factory { } - provide a factory bean definition
- single { } - provide a bean definition
- get() - resolve a component dependency
- bind - additional Kotlin type binding for given bean definition
- getProperty() - resolve a property

```
    // Koin for Android
    implementation 'org.koin:koin-android:1.0.1'

    implementation 'org.koin:koin-android-viewmodel:1.0.1'

    testImplementation 'org.koin:koin-test:1.0.1'
```

Koin DSL

```
class ServiceB()
class ServiceA(val serviceB: ServiceB)
class ServiceC(val serviceA: ServiceA, val serviceB: ServiceB)

val myModule = module { this: ModuleDefinition

    single { ServiceB() }

    factory { ServiceA(get()) }

    module( path: "subModule") { this: ModuleDefinition
        single { ServiceC(get(), get()) }
    }
}
```

```
class MyActivity : AppCompatActivity() {

    val serviceA: ServiceA by inject()

}
```

Koin and Architecture Components

```
class DetailActivity : AppCompatActivity() {

    val detailViewModel by viewModel<DetailViewModel>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_weather_detail)

        detailViewModel.uiData.observe(this, android.arch.lifecycle.Observer { uiData ->

            // observe data …
        })

    }
}
```

```
class DetailViewModel(val weatherRepository: WeatherRepository) : ViewModel() {

    val uiData = MutableLiveData<DailyForecastModel>()

    fun getDetail(id: String) {

        // get data with weatherRepository ...
    }
}
```

```
val weatherModule = applicationContext {

    // Declare DetailViewModel
    viewModel { DetailViewModel(get()) }

    // Declare WeatherRepository singleton
    bean { WeatherRepositoryImpl(get()) as WeatherRepository }
}
```

Sharing ViewModel?

```
class ResultActivity : AppCompatActivity() {

    // Declare ViewModel in Activity
     val model: WeatherResultViewModel by viewModel()

}
```

```
class ResultListFragment : Fragment() {

    // Shared ViewModel with parent Activity
     val model: WeatherResultViewModel by sharedViewModel()

}
```

Testing

```
class DetailViewModelTest : KoinTest {

    val viewModel: DetailViewModel by inject()
    val repository: WeatherRepository by inject()

    @Mock
    lateinit var uiData: Observer<DailyForecastModel>

    @get:Rule
    val rule = InstantTaskExecutorRule()

    @Before
    fun before() {
        MockitoAnnotations.initMocks(this)
        startKoin(testApp)
    }
```

```kotlin
@Test
fun testGotDetail() {
    // Setup data
    repository.searchWeather("Toulouse").blockingGet()
    val list = repository.getWeather().blockingGet()

    // Observe
    viewModel.uiData.observeForever(uiData)

    // Select data to notify
    val weather = list.first()
    viewModel.getDetail(weather.id)

    // Has received data
    Assert.assertNotNull(viewModel.uiData.value)

    // Has been notified
    Mockito.verify(uiData).onChanged(weather)
}
```

Demo

References

- https://insert-koin.io/
- https://android.jlelse.eu/unlock-your-android-viewmodel-power-with-koin-23eda8f493be
- https://github.com/anujmiddha/github-sample

Thanks!