devfest

抖音创作工具 Android 架构思考与实践

王鹏 Android GDE

Google Developer Groups

Tianjin

内容

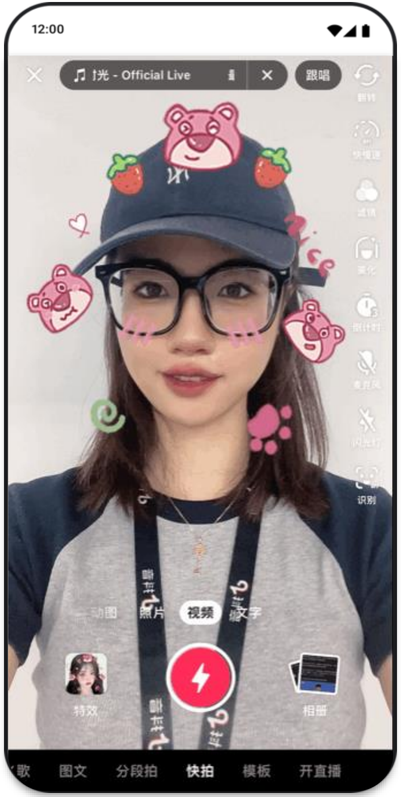
❖ 业务特点及挑战

❖ 架构及其演进

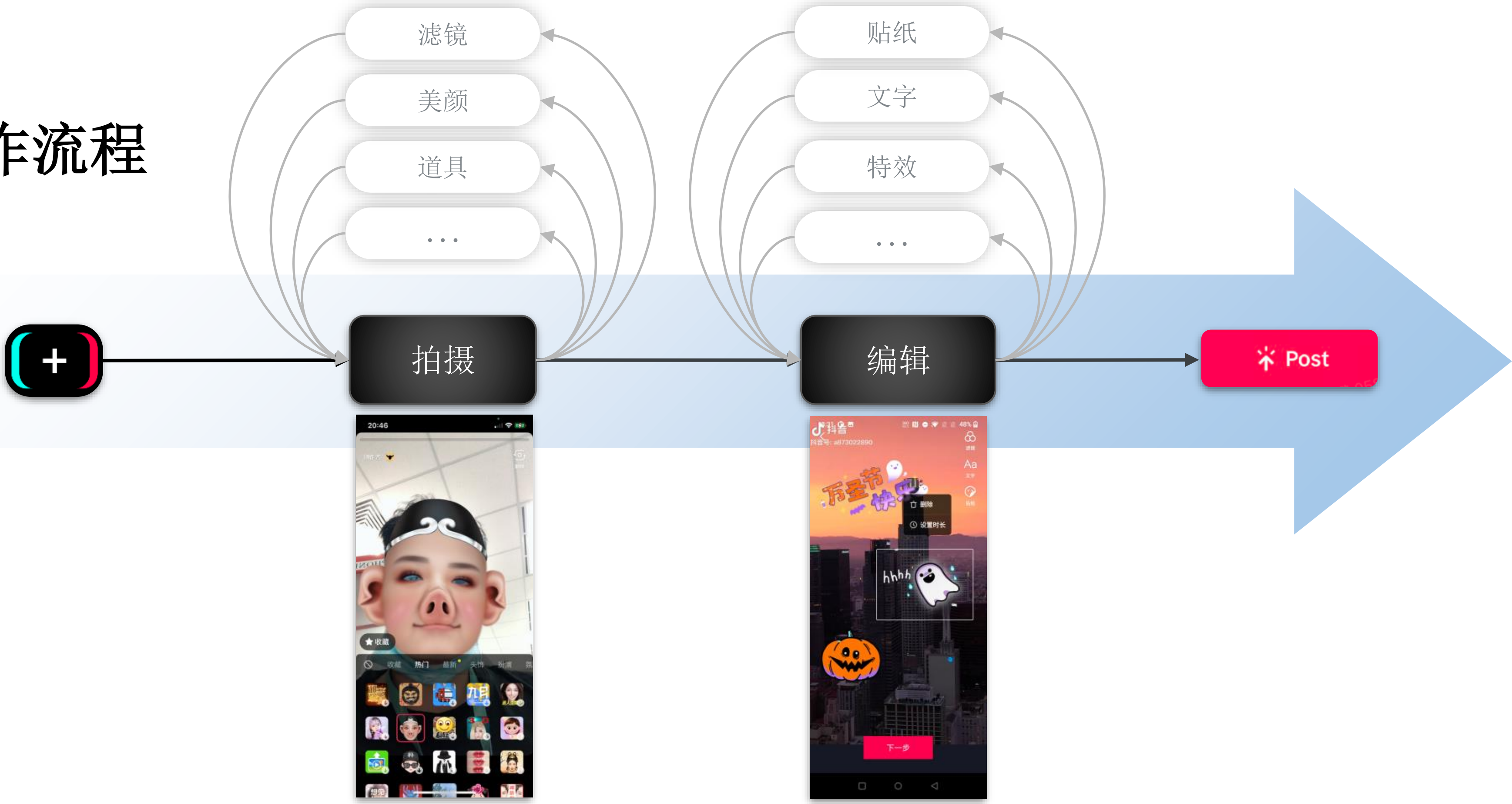
❖ 防劣化能力

创作工具

帮助用户拍摄和发布短视频作品，并通过特效、贴纸等有趣功能激发创意表达。



创作流程



滤镜
美颜
道具
. . .
贴纸
文字
特效
. . .
+
拍摄
编辑
Post
20:46
收藏
热门
最新
万圣节快乐
删除
设置时长
Aa
hhhh
下一步

业务挑战

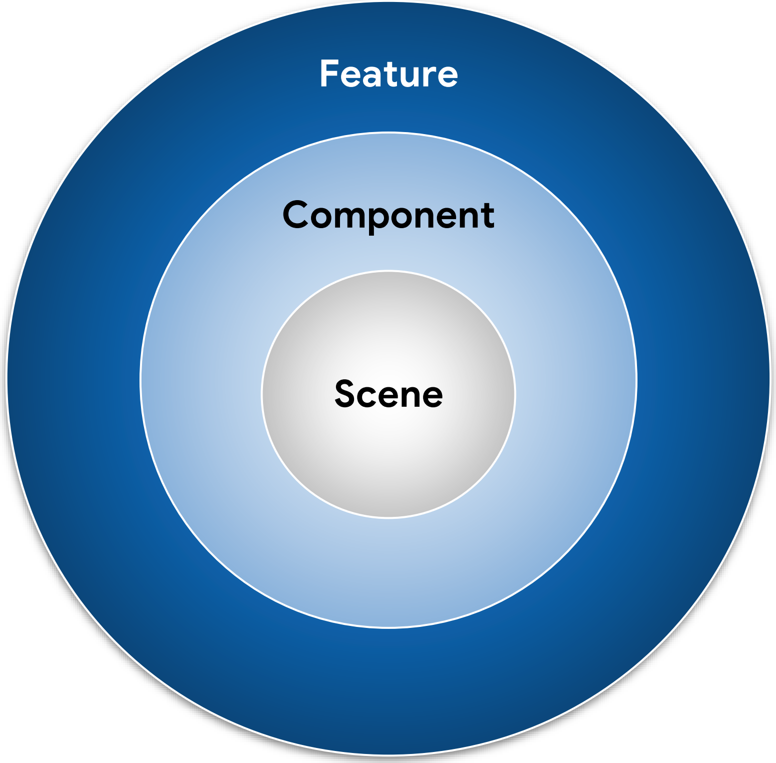
- 页面功能复杂，手势、面板、控件...
- ABTest驱动，功能替换频繁
- 性能敏感，首帧体验尤为关键

架构思考

- 页面组件化：快速搭建UI
- 关注点分离：UI&Logic解耦
- 按需加载：极致首帧体验
- 防劣化：及时发现架构问题

架构关键角色

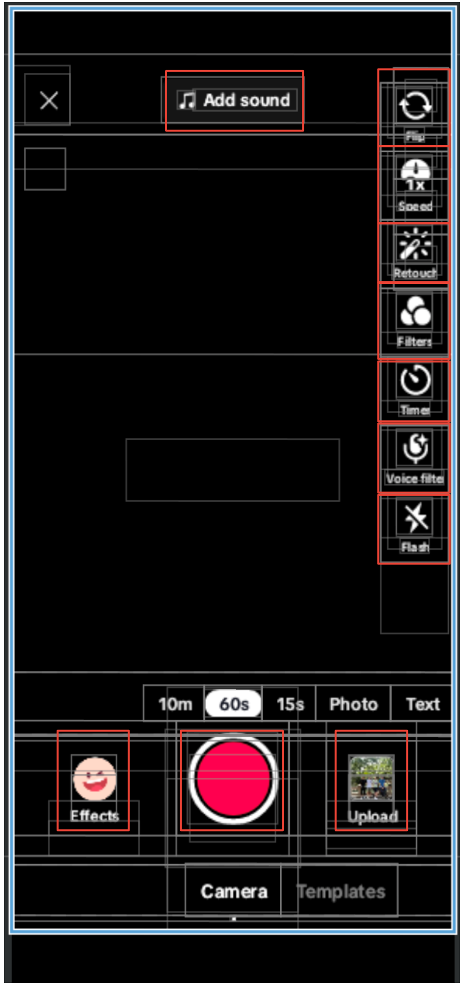
- ❖ Scene - UI单元
- ❖ Component - Logic单元
- ❖ Feature - 业务单元



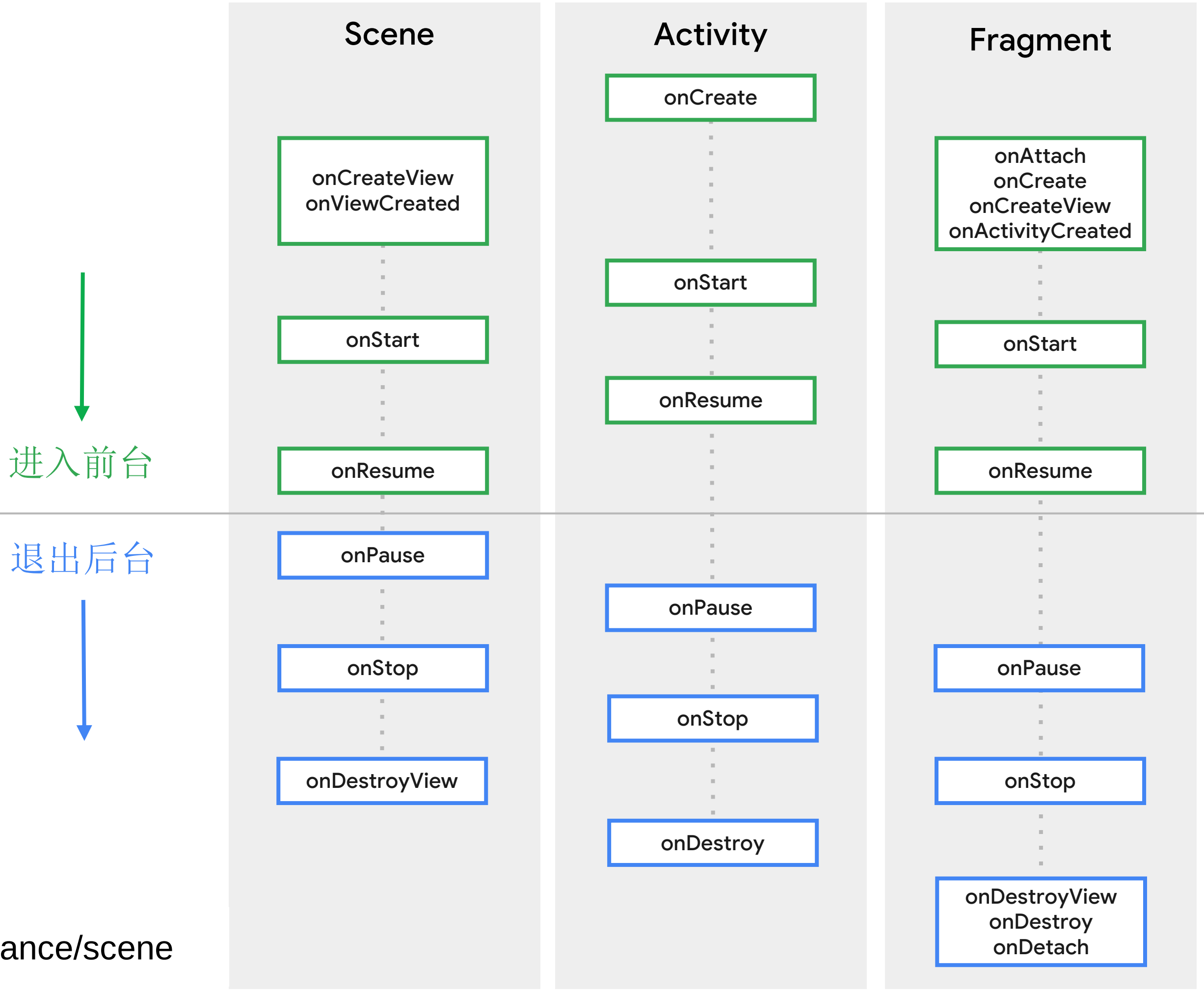
Scene

- 提供页面组合和导航，类比Fragment
- 规避Fragment的易用性问题

```kotlin
val childScene = ChildScene()
parentScene.add(R.id.container, childScene, "childScene")
parentScene.hide(childScene)
parentScene.show(childScene)
parentScene.remove(childScene)
```



Scene Lifecycle



进入前台

退出后台

Scene

onCreateView
onViewCreated

onStart

onResume

onPause

onStop

onDestroyView

Activity

onCreate

onStart

onResume

onPause

onStop

onDestroy

Fragment

onAttach
onCreate
onCreateView
onActivityCreated

onStart

onResume

onPause

onStop

onDestroyView
onDestroy
onDetach

https://github.com/bytedance/scene

Scene 的问题

- ❖ 创建/组装/显隐控制造成父容器逻辑复杂
- ❖ 内部Logic&UI无法解耦

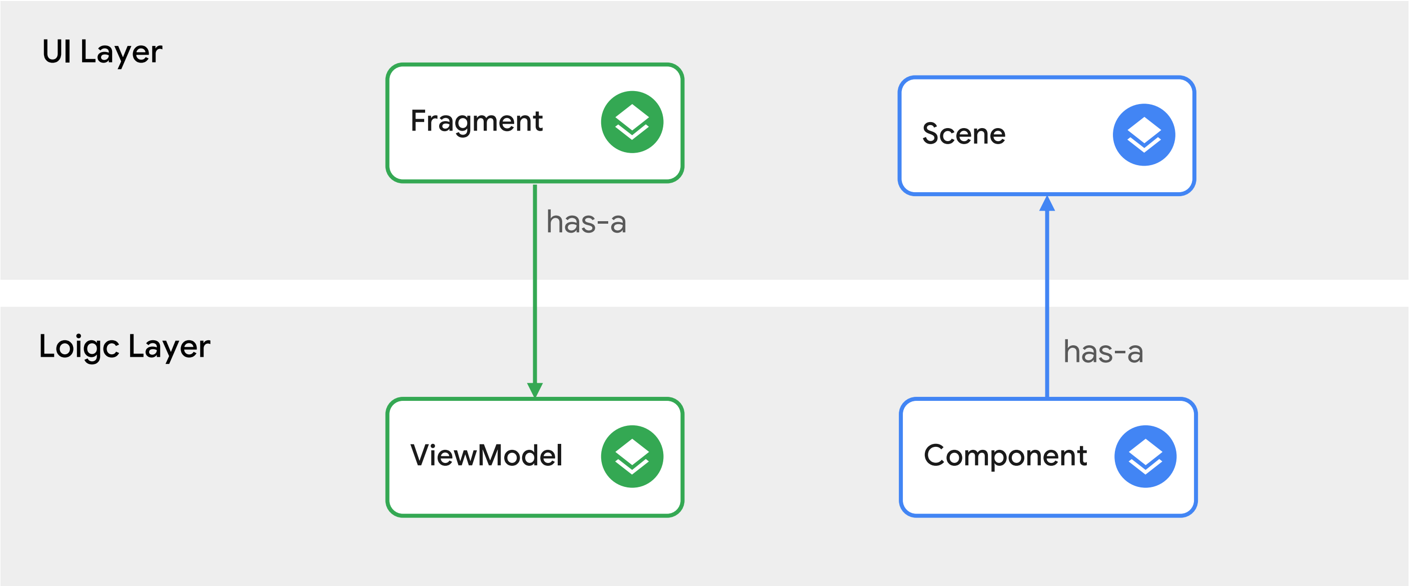
引入 Component

- 比Scene更长的生命周期
- 降低父容器负担
- 承载逻辑和状态
- 可按需加载

Component 创建 Scene

```kotlin
class FilterPanelComponent(private val parentScene: GroupScene) {
    //The scene created and holded in this component
    private val panel: Scene by lazy { ... }

    fun show(visible: Boolean) {
        if (!parentScene.contains(scene)) {
            parentScene.add(R.id.scene_container, panel, "filterPanel")
        }
        if(visible) parentScene.show(panel)
        else parentScene.hide(panel)
    }
}
```

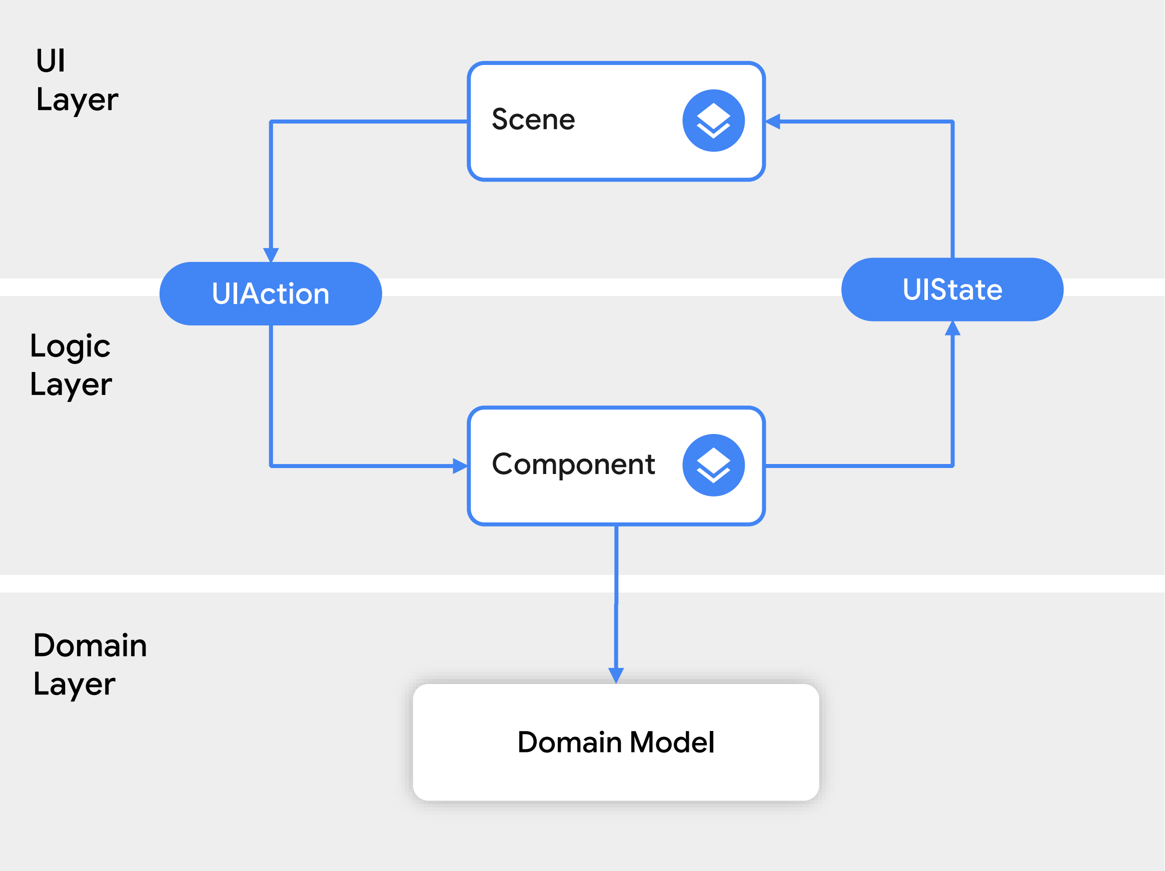
Component vs ViewModel



UDF 通信

“ A unidirectional data flow (UDF) is a design pattern where state flows down and events flow up. By following unidirectional data flow, you can decouple composables that display state in the UI from the parts of your app that store and change state. ”

https://developer.android.com/jetpack/compose/architecture#udf



```kotlin
class MyComponent(private val parentScene: GroupScene) :
    UIComponent<MySceneStates, MySceneActions>() {

    //构建 UI 初始状态
    override val defaultStates = {
        MySceneStates(
            name = "default"
        )
    }

    //响应 UI 事件
    override val sceneActions = {
        MySceneActions(
            onButtonClicked = this::onButtonClicked,
        )
    }
}
```

```kotlin
data class MySceneStates(
    val name: String
)

data class MySceneActions(
    val onButtonClicked: () -> Unit
)
```

```kotlin
private fun onButtonClicked() {
    ...
    setState {
        //this: MySceneStates
        copy(name = "newValue")
    }
}
```

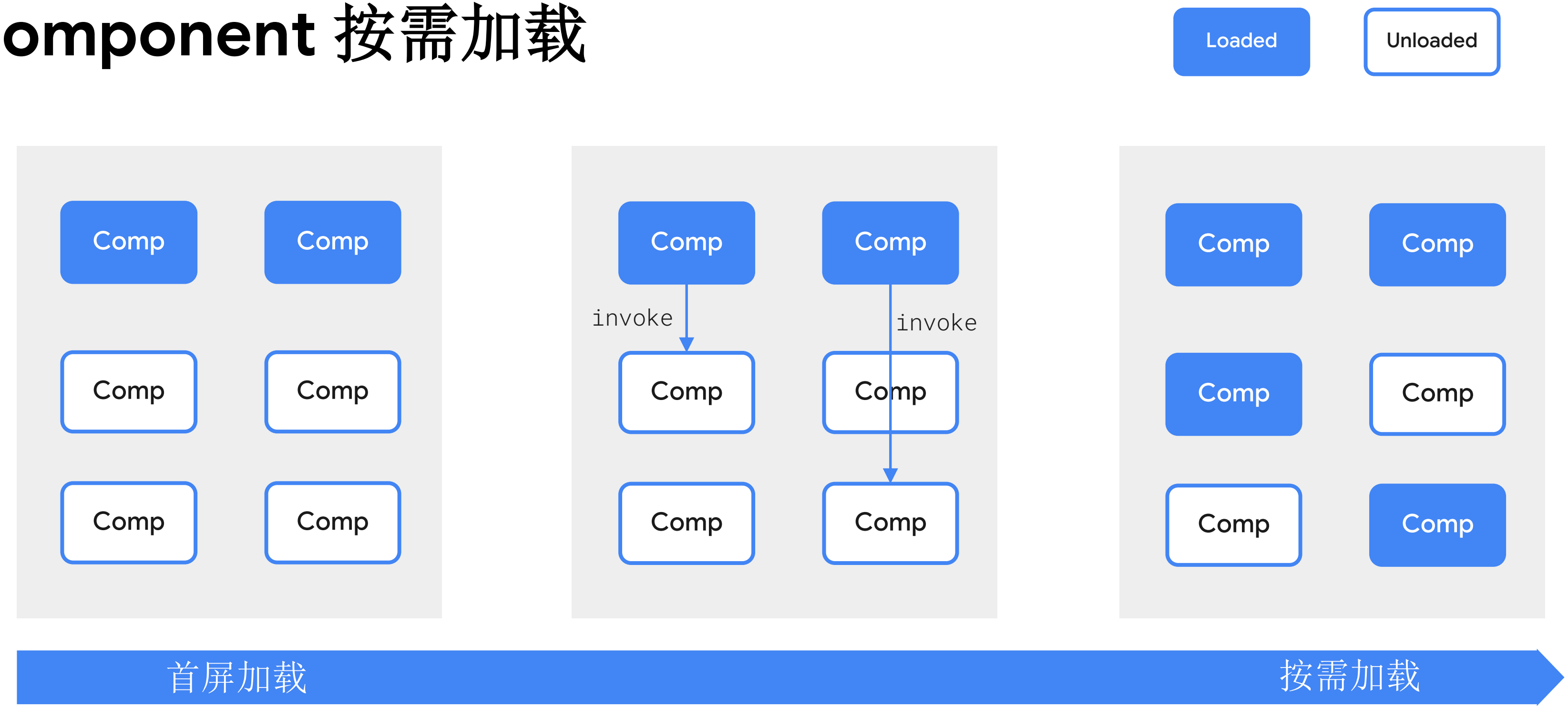
https://github.com/airbnb/mavericks

```kotlin
class MyScene : UIScene<MySceneStates, MySceneActions>() {
    private lateinit var textView: TextView
    private lateinit var button: Button

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)
        //observe state change
        uiStates.observe(MySceneStates::name) {
            textView.text = it
        }
        button.setOnClickListener {
            uiActions.onButtonClicked()
        }
        //read state value
        val name = uiStates.value.name
    }
}
```

DSL 组装 Component

```
class ComponentActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        components {
            //registe components using DSL
            component { ToolbarComponent(...) }
            component { FilterPanelComponent(...) }
            component { BeautyPanelComponent(...) }
            ...
        }
    }
}
```

Component 按需加载



Google Developer Groups

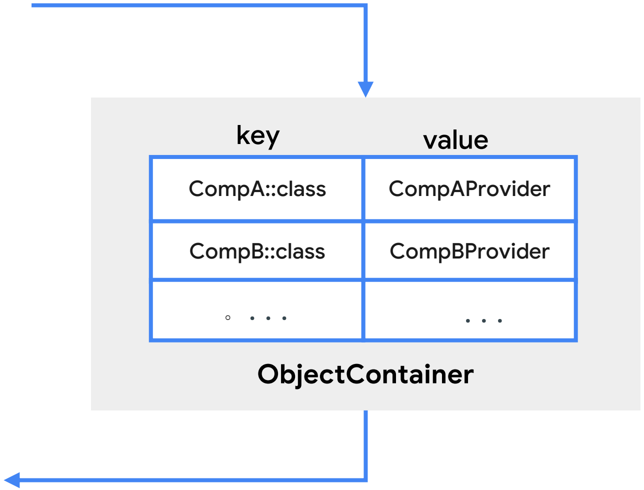
Component 按需加载

```
components {
    component { ToolbarComponent(...) }
    component(AttachOption.LAZY) { FilterPanelComponent(...) } //按需加载
    component(AttachOption.LAZY) { BeautyPanelComponent(...) } //按需加载
    ...
}
```

```
class ToolbarComponent()  {
    private val FilterPanelComponent: FilterPanelComponent by inject()

    fun showFilterPanel() {
        FilterPanelComponent.showPanel()
    }
}
```

Component 服务发现

```kotlin
@ComponentDsl
inline fun <reified T> component(
    attachOption: AttachOption = AttachOption.IMMEDIATE
    crossinline init: (ComponentBuilder).(ObjectContainer) -> T
): ComponentBuilder
```



key	value
CompA::class	CompAProvider
CompB::class	CompBProvider
◦ . . .	. . .

ObjectContainer

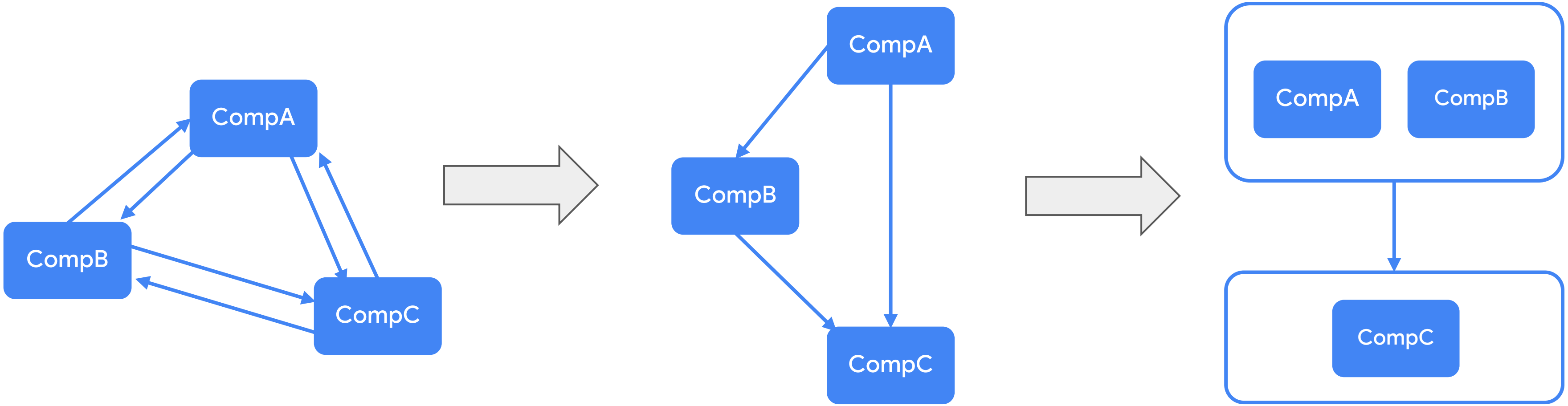
```kotlin
inline fun <reified T> ObjectContainer.inject(): Lazy<T> =
    lazy(LazyThreadSafetyMode.NONE) {
        val compProvider = this.getProvider<T>(T::class.java)
        compProvider.get()
    }
```

https://insert-koin.io/

Component 的问题

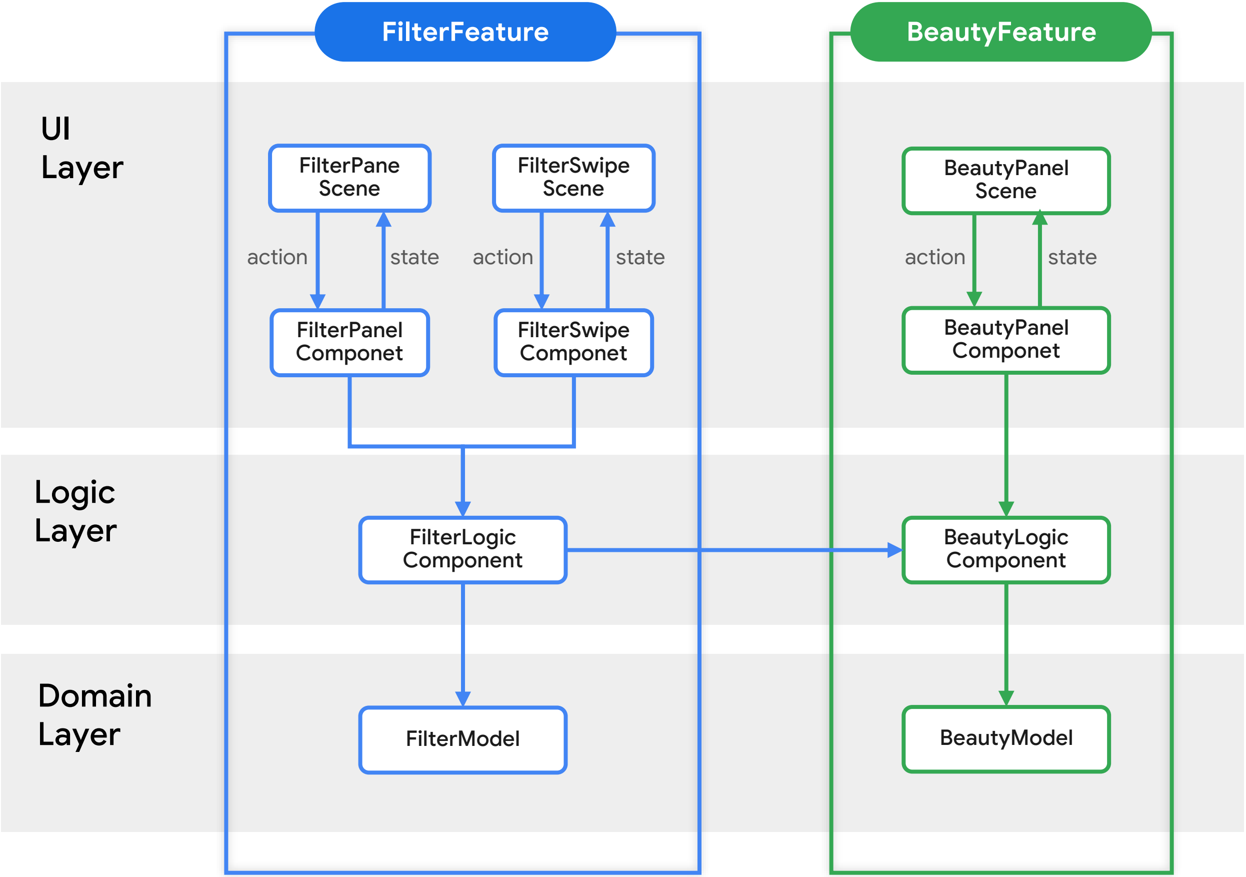
"*the dependency graph of packages or components should have no cycles*"

- 《Clean Architecture》



Feature

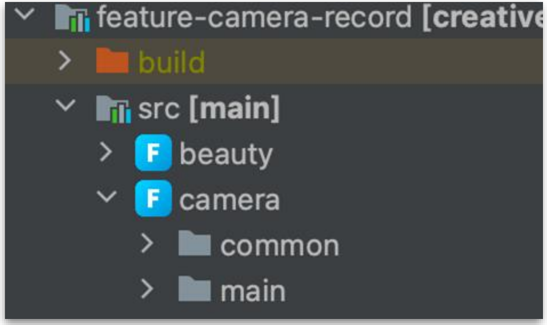
- ❖ Component集合
- ❖ 高内聚、低耦合
- ❖ 单向依赖
- ❖ 垂直分层



Google Developer Groups

Feature Module

- SourceSet组织目录
- IDE-Plugin截断代码联想以及代码/资源隔离飘红
- Gradle配置依赖关系



```
feature-camera-record [creative
  build
  src [main]
    beauty
    camera
      common
      main
```

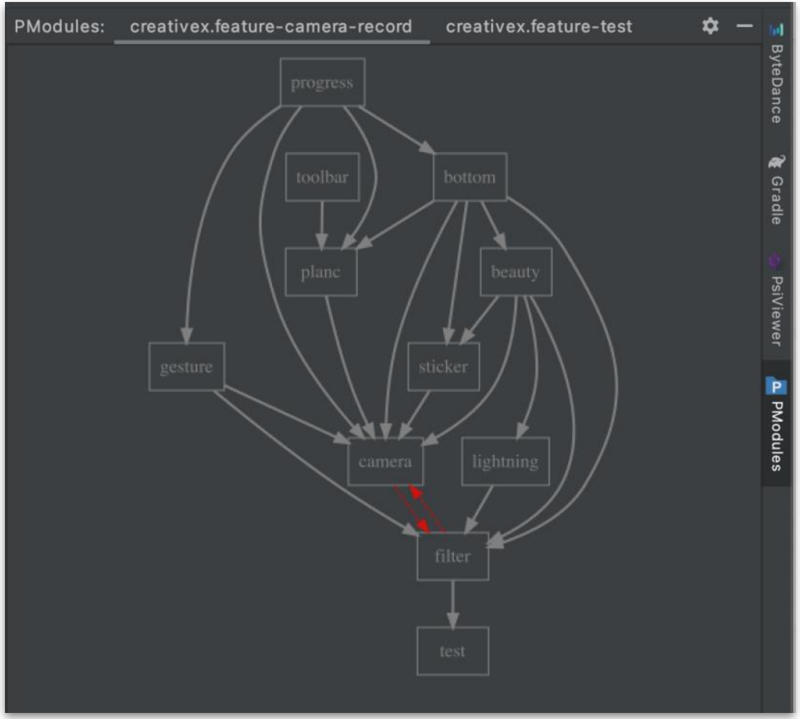
IDE 呈现

```
src
├── beauty
│   ├── open #外部可访问
│   │   └── BeautyLogicComp
│   └── panel
│       ├── BeautyPanelScene
│       ├── BeautyPanelUiState
│       └── BeautyPanelComp
│
├── filter
│   ├── panel
│   │   ├── FilterPanelScene
│   │   ├── FilterPanelUiState
│   │   └── FilterPanelComp
│   ├── swpie
│   │   ├── FilterSwipeScene
│   │   ├── FilterSwipeUiState
│   │   └── FilterSwipeComp
│   └── logic
│       └── FilterLogicComp
│
├── camera
│   ├── open
│   │   └── CameraComponent
│   └── ...
│
├── ...
```

Google Developer Groups

features.gradle

```
featureModules {
    filter {
        path "com.douyin.tools.filter"
        dependsOn "camera", "beauty"
    }
    camera {
        path "com.douyin.tools.camera"
        dependsOn "filter"
    }
    beauty {
        path "com.douyin.tools.beauty"
        dependsOn "camera", "sticker"
    }
    sticker {
        path "com.douyin.tools.sticker"
        dependsOn "camera"
    }
}
```



依赖关系可视化

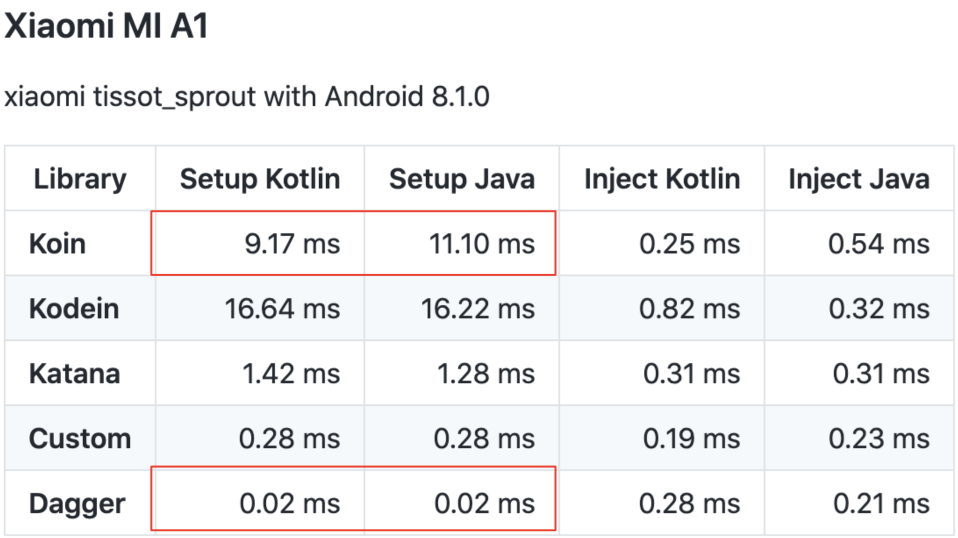
Component 服务发现的问题

- Component构造成本高
- 依赖缺失造成运行时异常

```
components {
    component { ToolbarComponent(...) }
    component(AttachOption.LAZY) { FilterPanelComponent(...) }
    component(AttachOption.LAZY) { BeautyPanelComponent(...) }
    ...
}
```

Dagger over Koin

- 效率：降低容器复杂度
- 质量：编译期发现问题
- 性能：降低启动开销



Xiaomi MI A1

xiaomi tissot_sprout with Android 8.1.0

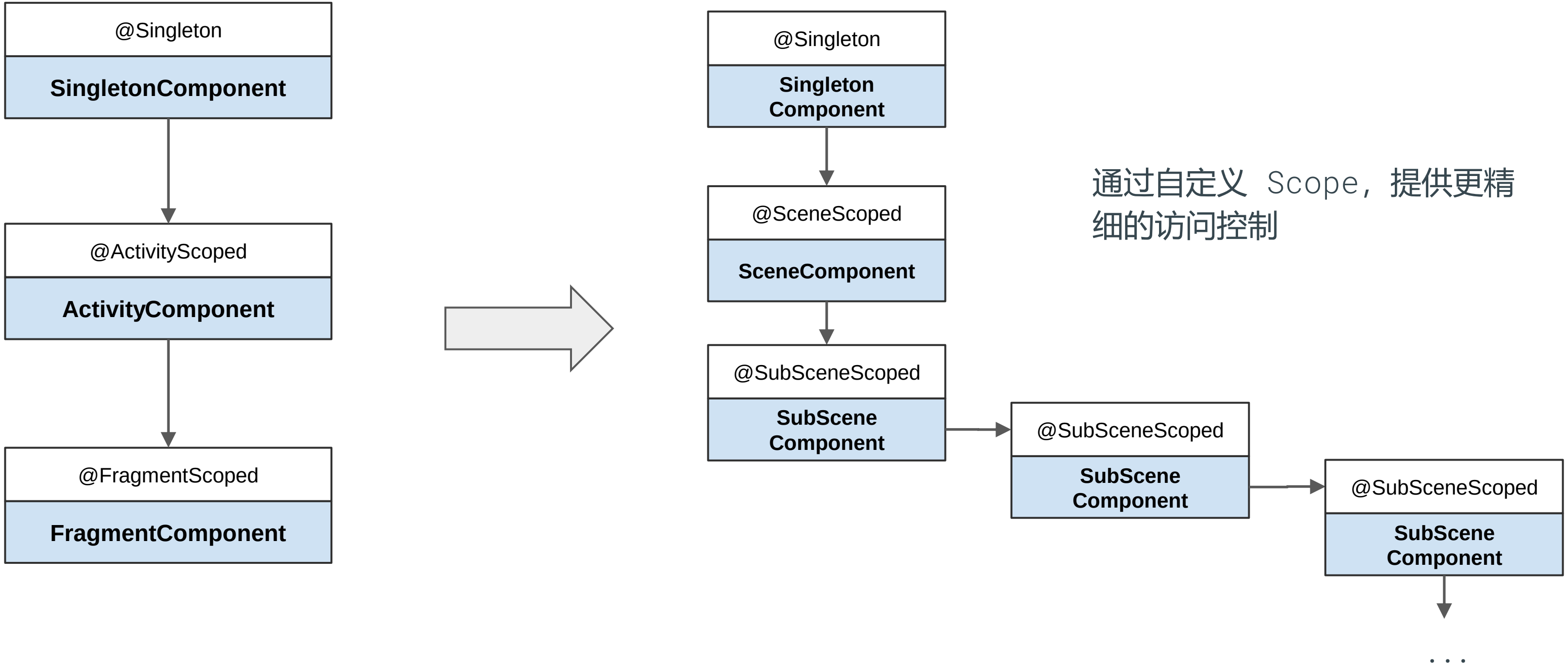
Library	Setup Kotlin	Setup Java	Inject Kotlin	Inject Java
Koin	9.17 ms	11.10 ms	0.25 ms	0.54 ms
Kodein	16.64 ms	16.22 ms	0.82 ms	0.32 ms
Katana	1.42 ms	1.28 ms	0.31 ms	0.31 ms
Custom	0.28 ms	0.28 ms	0.19 ms	0.23 ms
Dagger	0.02 ms	0.02 ms	0.28 ms	0.21 ms

引入 DI

```
components {
    component(R.id.xxx, CameraComponent::class)
    component(AttachOption.LAZY, R.id.xxx, FilterPanelComponent::class)
    component(AttachOption.LAZY, R,id,xxx, BeautyPanelComponent::class)
    ...
}
```

```
@Component //generate inject code
class FilterLogicComponent @Inject constructor() {

    @Inject lateinit var filterModel: FilterModel
    val beautyLogicComponent: BeautyLogicComponent by inject()


    fun onCreate() {
        filterModel.doSth() // crash by invalid invoke
    }
}
```

自定义 Scope

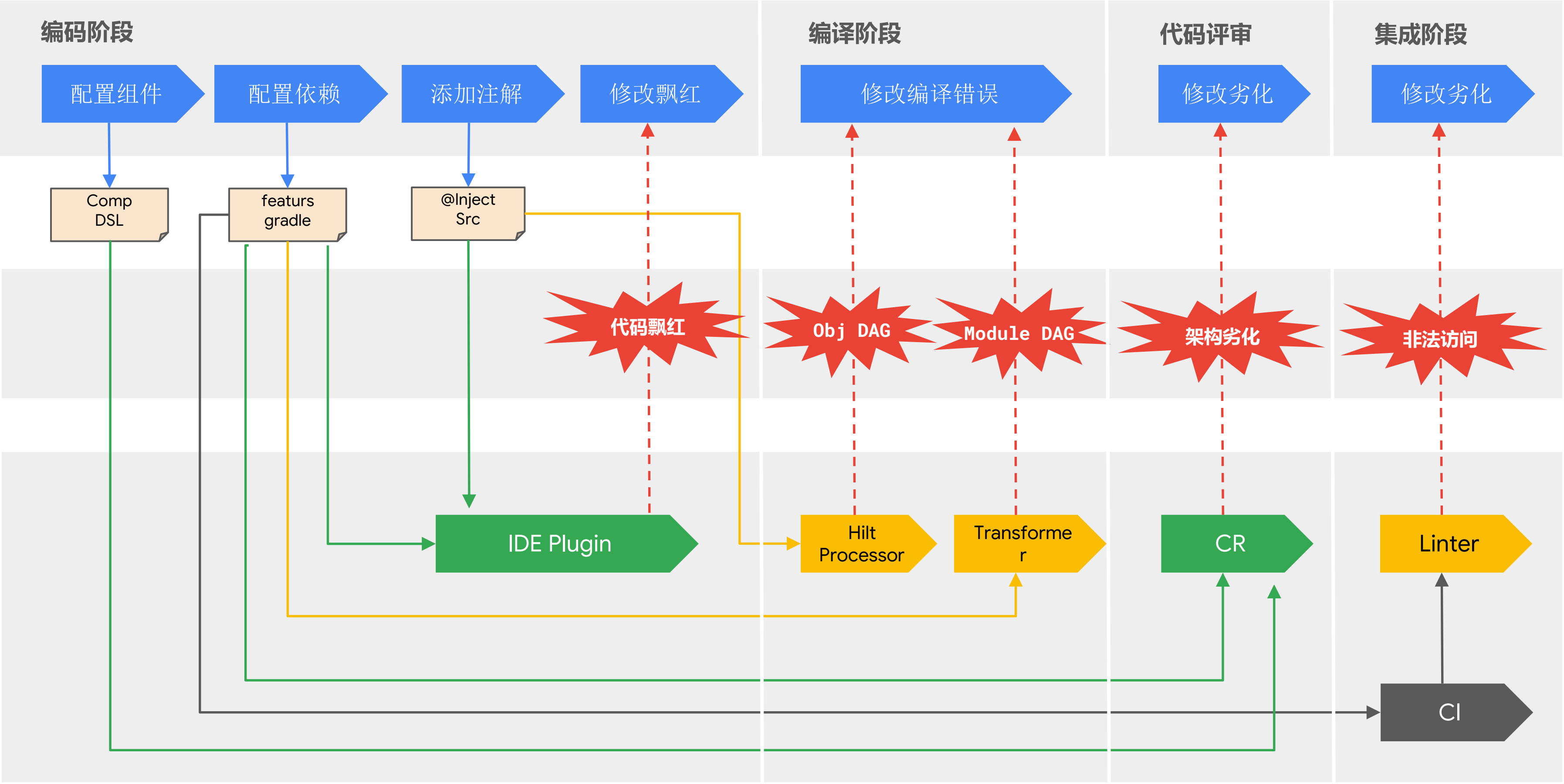


通过自定义 Scope，提供更精细的访问控制

自定义 Scope

```
@DefineContainer
annotation class RootContainer

@DefineContainer(parent = RootContainer::class) // build scope relationship
class FilterContainer

@Container(FilterContainer::class)
class FilterScene : GroupScene() { ... }


@Component(container = FilterContainer::class)  //generate inject code
class FilterLogicComponent { ... }


@Module
@InstallIn(FilterContainer::class) //provide module to container
interface FilterModule { ... }
```

防劣化能力



编码阶段
编译阶段
代码评审
集成阶段
配置组件
配置依赖
添加注解
修改飘红
修改编译错误
修改劣化
修改劣化
Comp DSL
featurs gradle
@Inject Src
代码飘红
Obj DAG
Module DAG
架构劣化
非法访问
IDE Plugin
Hilt Processor
Transforme r
CR
Linter
CI

总结

从现状出发，参考 MAD、Mvrx、Koin、Hilt 等优秀开源项目，摸索出了适合创作工具业务的最佳实践：

- ❖ Scene - 拆分页面 UI
- ❖ Component - 功能配置、承载逻辑
- ❖ Featue - 实现业务模块的合理依赖
- ❖ DI - 实现组件通信和依赖构建

devfest

```
// You'll need
// com.google.
listRef.listAl
       .addOn
          prefixes.fo
            // All
            // You
          }
          it      ach { item
                   the items
          }
       }
```

Thank You！

加V交流

Google Developer Groups

Tianjin