요약

Component

Dagger2

ApplicationComponent

```
@Singleton
@Component(modules = [AndroidSupportInjectionModule::
class, NetworkModule::class, ActivityBuilder::class])
interface ApplicationComponent {
    @Component.Builder
    interface Builder {
        @BindsInstance
        fun application(application: Application):
Builder

        fun build(): ApplicationComponent
    }

    fun inject(app: RootApplication)
}
```

Hilt

컴포넌트 제공(작성 X)

SingletonComponent,

ActivityRetainedComponent,

ActivityComponent,

FragmentComponent,

ViewComponent,

ViewWithFragmentComponent,

ServiceComponent

참조 : Component hierarchy

Module

Dagger2

NetworkModule

```
@Module
class NetworkModule {

    @Provides
    @Singleton
    fun provideApiService(okHttpClient:
OkHttpClient): ApiService {
        ...
    }

    @Provides
    @Singleton
    fun provideRepository(apiService: ApiService):
Repository = ImageRepository(apiService)
}
```

Hilt

NetworkModule

```
@Module
@InstallIn(SingletonComponent::class)
object NetworkModule {

    @Singleton
    @Provides
    fun provideApiService(okHttpClient:
OkHttpClient): ApiService {
        ...
    }

    @Singleton
    @Provides
    fun provideRepository(apiService:
ApiService): Repository = ImageRepository
(apiService)
}
```

Application

Dagger2

Application

```
class RootApplication: Application(),
HasActivityInjector {
    @Inject
    lateinit var activityDispatchingAndroidInjector:
DispatchingAndroidInjector<Activity>

    override fun activityInjector():
AndroidInjector<Activity> =
activityDispatchingAndroidInjector

    override fun onCreate() {
        super.onCreate()
        DaggerApplicationComponent.builder()
                        .application(this)
                        .build()
                        .inject(this)
    }
}
```

Hilt

Application 구현

```
@HiltAndroidApp
class RootApplication: Application() {
    ...
}
```

Vie
wM
odel

ViewModel

```
class MainViewModel(val repository: Repository) :
ViewModel() {

    @SuppressLint("CheckResult")
    fun fetchImages(query: String, page: Int, size:
Int) {
        repository.fetchImages(query, page, size)
        ...
    }
}
```

ViewModelFactory

```
class MainViewModelFactory @Inject constructor
(private val repository: Repository) :
ViewModelProvider.Factory {
    override fun <T : ViewModel?> create(modelClass:
Class<T>): T {
        return MainViewModel(repository) as T
    }
}
```

ViewModel

```
@HiltViewModel
class MainViewModel @Inject constructor(
        val repository: Repository
): ViewModel() {

    fun fetchImages(searchText: String) {
        repository.fetchImages(searchText)
        ...
    }
}
```

Acti
vity

Activity

```
class MainActivity : AppCompatActivity() {
    @Inject
    lateinit var viewModelFactory:
MainViewModelFactory

    private val mainViewModel: MainViewModel by lazy{
        ViewModelProviders.of(this, viewModelFactory)
[MainViewModel::class.java]
    }
}
```

Activity

```
@AndroidEntryPoint
class MainActivity : AppCompatActivity() {
    private val viewModel by
viewModels<MainViewModel>()
    ...
}
```