# Hilt & Dagger 注解

v2.33 - @AndroidDev

| 注解 | 用途 | 代码示例 |
| --- | --- | --- |
| @HiltAndroidApp | 启用 Hilt 代码生成。<br>必须注解 Application 类。 | `@HiltAndroidApp`<br>`class MyApplication : Application() { ... }` |
| @AndroidEntryPoint | 将 DI 容器添加到被注解的 Android 类中。这需要用到 Hilt 的 Gradle 插件。 | `@AndroidEntryPoint`<br>`class MyActivity : AppCompatActivity() { ... }` |
| @Inject | 构造方法注入：注明哪些构造方法可以用来提供实例以及该类型包含哪些依赖关系。 | `class AnalyticsAdapter @Inject constructor(`<br>`  private val service: AnalyticsService`<br>`) { ... }` |
| | 字段注入：为被 @AndroidEntryPoint 注解的类中的字段赋值。<br>字段不能是私有的。 | `@AndroidEntryPoint`<br>`class MyActivity : AppCompatActivity() {`<br>`  @Inject lateinit var adapter: AnalyticsAdapter`<br>`  ...`<br>`}` |
| @HiltViewModel | 注明 Hilt 如何提供一个组件架构 ViewModel 的实例。 | `@HiltViewModel`<br>`class MyViewModel @Inject constructor(`<br>`  private val adapter: AnalyticsAdapter,`<br>`  private val state: SavedStateHandle`<br>`): ViewModel() { ... }` |
| @Module | 在被注解的类中，您可以为无法使用构造方法注入的类型添加绑定。 | `@InstallIn(SingletonComponent::class)`<br>`@Module`<br>`class AnalyticsModule { ... }` |
| @InstallIn | 标明模块绑定必须在哪些 Hilt 生成的 DI 容器中可用（此处为代码中的 `SingletonComponent` ）。 | `@InstallIn(SingletonComponent::class)`<br>`@Module`<br>`class AnalyticsModule { ... }` |
| @Provides | 为无法进行构造方法注入的类型添加绑定：<br>- 返回类型是该绑定类型；<br>- 参数为依赖项；<br>- 如果类型没有被限定作用域，每次需要实例时，都会执行该方法。 | `@InstallIn(SingletonComponent::class)`<br>`@Module`<br>`class AnalyticsModule {`<br>`  @Provides`<br>`  fun providesAnalyticsService(`<br>`  converterFactory: GsonConverterFactory`<br>`  ): AnalyticsService {`<br>`  return Retrofit.Builder()`<br>`  .baseUrl("https://example.com")`<br>`  .addConverterFactory(converterFactory)`<br>`  .build()`<br>`  .create(AnalyticsService::class.java)`<br>`  }`<br>`}` |
| @Binds | 绑定接口类型速记：<br>- 方法必须处于被 @Module 注解的类中；<br>- 被 @Bind 注解的方法必须是抽象的；<br>- 返回类型是绑定的类型；<br>- 参数是该类型的实现。 | `@InstallIn(SingletonComponent::class)`<br>`@Module`<br>`abstract class AnalyticsModule {`<br>`  @Binds`<br>`  abstract fun bindsAnalyticsService(`<br>`  analyticsServiceImpl: AnalyticsServiceImpl`<br>`  ): AnalyticsService`<br>`}` |
| 作用域注解：<br>@Singleton<br>@ActivityScoped<br>... | 将对象的作用域限定为一个容器：<br>当字段注入或下层容器需要使用一个类型作为依赖项时，容器将提供该类型的同一个实例。 | `@Singleton`<br>`class AnalyticsAdapter @Inject constructor(`<br>`  private val service: AnalyticsService`<br>`) { ... }` |
| 预定义绑定限定符：<br>@ApplicationContext<br>@ActivityContext | 预定义绑定可以在对应的容器中当作依赖项使用。<br>这些是限定符注解。 | `@Singleton`<br>`class AnalyticsAdapter @Inject constructor(`<br>`  @ApplicationContext val context: Context`<br>`  private val service: AnalyticsService`<br>`) { ... }` |
| @Entry Point | 获取既不能被 Hilt 直接支持，又不能使用 Hilt 的类的依赖项。<br>被 @EntryPoint 注解的接口必须同时注解 @InstallIn，以传递从 @InstallIn 中获取依赖关系的 Hilt 预定义组件。<br>使用 EntryPointAccessors 类中适当的静态方法访问绑定，并传入 DI 容器（与 @InstallIn 注解中的值匹配）类的实例，以及被 @EntryPoint 注解的类对象。 | `class MyContentProvider(): ContentProvider {`<br>`  @InstallIn(SingletonComponent::class)`<br>`  @EntryPoint`<br>`  interface MyContentProviderEntryPoint {`<br>`  fun analyticsService(): AnalyticsService`<br>`  }`<br>`  override fun query(...): Cursor {`<br>`  val appContext =`<br>`  context?.applicationContext`<br>`  ?: throw IllegalStateException()`<br>`  val entryPoint =`<br>`  EntryPointAccessors.fromApplication(`<br>`  appContext,`<br>`  MyContentProviderEntryPoint::class.java)`<br>`  val analyticsService =`<br>`  entryPoint.analyticsService()`<br>`  ...`<br>`  }`<br>`}` |

有关 DI、Hilt 和 Dagger 的更多信息，请访问：https://developer.android.google.cn/dependency-injection