Uniwersytet Wrocławski

Rafał Lewandków

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

WYKŁAD 13

- Dagger-Hilt

Dagger-Hilt

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

aktywuje generowanie kodu Hilt, w tym klasę bazową dla Twojej aplikacji, która pełni rolę kontenera zależności na poziomie aplikacji

Dagger-Hilt

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

aktywuje generowanie kodu Hilt, w tym klasę bazową dla Twojej aplikacji, która pełni rolę kontenera zależności na poziomie aplikacji

```
@AndroidEntryPoint
class ExampleActivity : AppCompatActivity() { ... }
```

generuje indywidualny komponent Hilt

Dagger-Hilt

```
@HiltAndroidApp
class MyApp : Application() {
}
```

```
<application
    android:name=".MyApp"
    ...
```

Dagger-Hilt

```
@Module
@InstallIn(SingletonComponent::class)
object AppModule {}
```

- Adnotacja ta jest używana do oznaczania klas, które definiują jakie obiekty zależności powinny zostać utworzone i jak one powinny zostać połączone ze sobą.
- Klasy oznaczone adnotacją @Module to tzw. moduły, które służą do definiowania zależności i określają jakie obiekty powinny być tworzone w celu ich dostarczenia.

@InstallIn - deklaruje do których komponentów opisana klasa/obiekt powinna zostać dodana podczas generacji obiektów przez Hilt.

Dagger-Hilt

Hilt component	Injector for
`SingletonComponent`	`Application`
`ActivityRetainedComponent`	N/A
`ViewModelComponent`	`ViewModel`
`ActivityComponent`	`Activity`
`FragmentComponent`	`Fragment`
`ViewComponent`	`View`
`ViewWithFragmentComponent`	`View` annotated with `@WithFragmentBindings`
`ServiceComponent`	`Service`

Uniwersytet Wrocławski

Dagger-Hilt

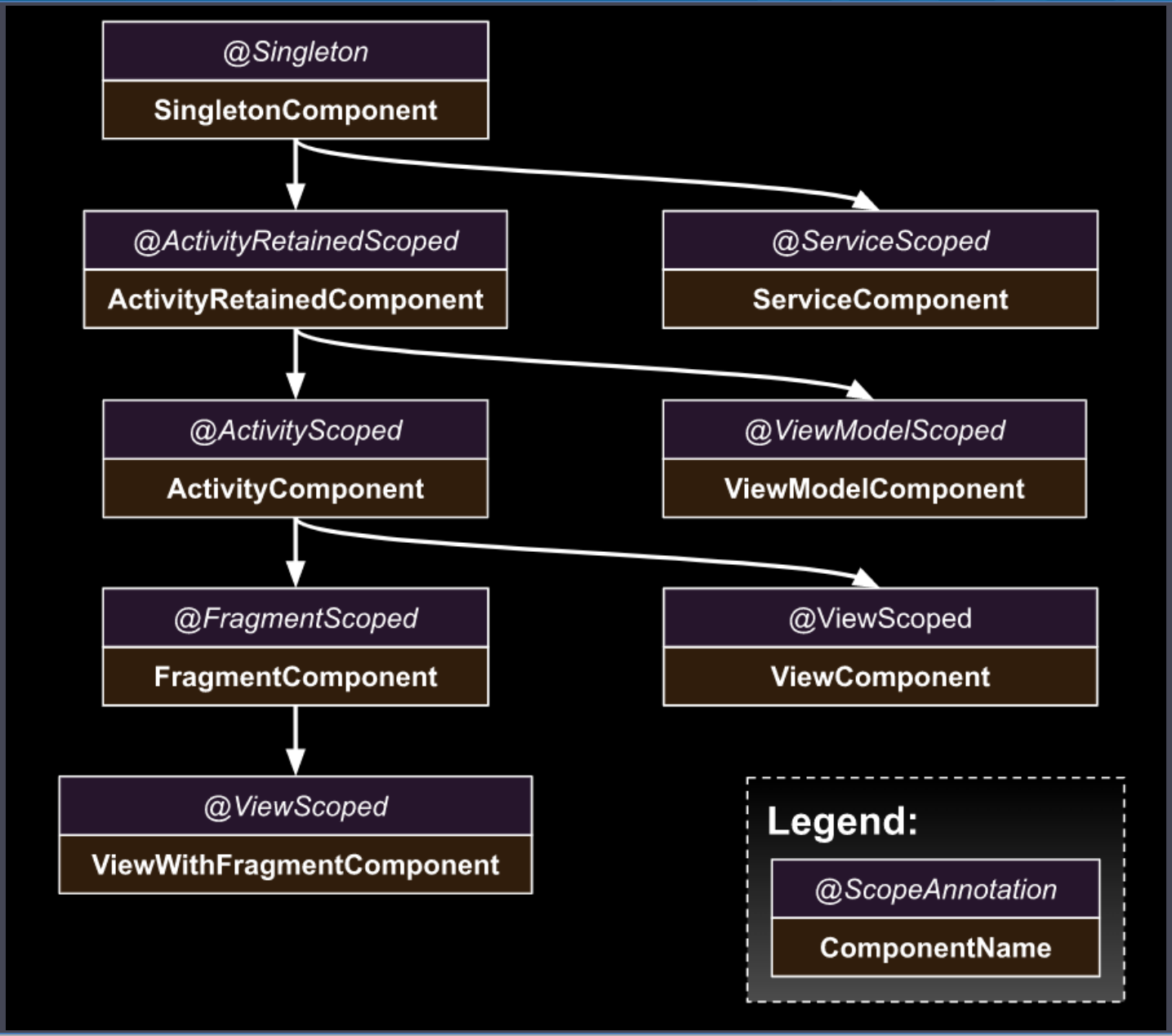
Android class	Generated component	Scope
`Application`	`SingletonComponent`	`@Singleton`
`Activity`	`ActivityRetainedComponent`	`@ActivityRetainedScoped`
`ViewModel`	`ViewModelComponent`	`@ViewModelScoped`
`Activity`	`ActivityComponent`	`@ActivityScoped`
`Fragment`	`FragmentComponent`	`@FragmentScoped`
`View`	`ViewComponent`	`@ViewScoped`
`View` annotated with `@WithFragmentBindings`	`ViewWithFragmentComponent`	`@ViewScoped`
`Service`	`ServiceComponent`	`@ServiceScoped`

Uniwersytet Wrocławski

Dagger-Hilt

Generated component	Created at	Destroyed at
`SingletonComponent`	`Application#onCreate()`	`Application` destroyed
`ActivityRetainedComponent`	`Activity#onCreate()`	`Activity#onDestroy()`
`ViewModelComponent`	`ViewModel` created	`ViewModel` destroyed
`ActivityComponent`	`Activity#onCreate()`	`Activity#onDestroy()`
`FragmentComponent`	`Fragment#onAttach()`	`Fragment#onDestroy()`
`ViewComponent`	`View#super()`	`View` destroyed
`ViewWithFragmentComponent`	`View#super()`	`View` destroyed
`ServiceComponent`	`Service#onCreate()`	`Service#onDestroy()`

Dagger-Hilt


@Singleton
SingletonComponent
@ActivityRetainedScoped
ActivityRetainedComponent
@ServiceScoped
ServiceComponent
@ActivityScoped
ActivityComponent
@ViewModelScoped
ViewModelComponent
@FragmentScoped
FragmentComponent
@ViewScoped
ViewComponent
@ViewScoped
ViewWithFragmentComponent
Legend:
@ScopeAnnotation
ComponentName

Dagger-Hilt

```
@Provides
@Singleton
fun provideAppRepository(api: PlaceholderApi) : AppRepository{
    return AppRepositoryImpl(api)
}
```

@Singleton - jeżeli PlaceholderApi zostanie wstrzyknięty do kilku klas, będzie to ta sama instancja. Bez tej adnotacji, przy wstrzykiwaniu do kilku klas, za każdym razem tworzona jest nowa instancja.

```
@HiltViewModel
class AppViewModel @Inject constructor (
    private val repository: AppRepository
) : ViewModel() {
```