Mastering Dagger with Hilt

on Android 

Myo Lwin Oo

Dependency injection is a technique in which an object receives other objects that it depends on.

DI frameworks for Android

- Dagger (https://dagger.dev/)
- Koin (https://insert-koin.io/)
- Kodein (https://kodein.org/Kodein-DI/)
- Guice (https://github.com/google/guice)

Dagger

Dagger
Adding dependency

```
apply plugin: 'kotlin-kapt'

dependencies {
    implementation "com.google.dagger:dagger:2.28.3"
    kapt "com.google.dagger:dagger-compiler:2.28.3"
}
```

Dagger

Create a component

```
@Component
interface ApplicationComponent { ... }

class MyApp: Application() {

    val appComponent: ApplicationComponent by lazy {
        DaggerApplicationComponent.create()
    }
}
```

Dagger

Create a component

```
@Component
interface ApplicationComponent { ... }

class MyApp: Application() {

    val appComponent: ApplicationComponent by lazy {
        DaggerApplicationComponent.create()
    }
}
```

Dagger

Create a component

```
@Component
interface ApplicationComponent { ... }

class MyApp: Application() {

    val appComponent: ApplicationComponent by lazy {
        DaggerApplicationComponent.create()
    }
}
```

Dagger

Create a component

```
@Component
interface ApplicationComponent {

    fun inject(activity: MainActivity)
}
```

Dagger
Inject into Activity

```
class MainActivity : AppCompatActivity() {

    @Inject lateinit var myAwesomeHelper: MyAwesomeHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        (applicationContext as MyApp).appComponent.inject(this)
        super.onCreate(savedInstanceState)

        myAwesomeHelper.doAwesomeThings()
    }
}
```

Dagger
Inject into Activity

```
class MainActivity : AppCompatActivity() {

    @Inject lateinit var myAwesomeHelper: MyAwesomeHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        (applicationContext as MyApp).appComponent.inject(this)
        super.onCreate(savedInstanceState)

        myAwesomeHelper.doAwesomeThings()
    }
}
```

Dagger

Inject into Activity

```
class MainActivity : AppCompatActivity() {

    @Inject lateinit var myAwesomeHelper: MyAwesomeHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        (applicationContext as MyApp).appComponent.inject(this)
        super.onCreate(savedInstanceState)

        myAwesomeHelper.doAwesomeThings()
    }
}
```

Dagger
Inject into Activity

```
class MainActivity : AppCompatActivity() {

    @Inject lateinit var myAwesomeHelper: MyAwesomeHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        (applicationContext as MyApp).appComponent.inject(this)
        super.onCreate(savedInstanceState)

        myAwesomeHelper.doAwesomeThings()
    }
}
```

Dagger
Prepare dependency

```
class MyAwesomeHelper @Inject constructor() {
    fun doAwesomeThings() {
        // this method does amazing things
    }
}
```

Dagger
Prepare dependency

```
class MyAwesomeHelper @Inject constructor() {
    fun doAwesomeThings() {
        // this method does amazing things
    }
}
```

Dagger

Setting up module for third-party classes

```
@Module
class NetworkModule {

    @Provides
    fun provideRetrofitService(): RetrofitService {
        return Retrofit.Builder()
            .baseUrl("https://example.com")
            .build()
            .create(RetrofitService::class.java)
    }
}
```

Dagger

Setting up module for third-party classes

```
@Module
class NetworkModule {

    @Provides
    fun provideRetrofitService(): RetrofitService {
        return Retrofit.Builder()
            .baseUrl("https://example.com")
            .build()
            .create(RetrofitService::class.java)
    }
}
```

Dagger

Setting up module for third-party classes

```
@Module
class NetworkModule {

    @Provides
    fun provideRetrofitService(): RetrofitService {
        return Retrofit.Builder()
            .baseUrl("https://example.com")
            .build()
            .create(RetrofitService::class.java)
    }
}
```

Dagger

Connect module with component

```
@Component(modules = [NetworkModule::class])
interface ApplicationComponent {
    ...
}
```

Dagger

Connect module with component

```
@Component(modules = [NetworkModule::class])
interface ApplicationComponent {
    ...
}
```

Hilt

Hilt

Adding dependency

build.gradle (project)

```
buildscript {
    ...
    dependencies {
        ...
        classpath 'com.google.dagger:hilt-android-gradle-plugin:2.28-alpha'
    }
}
```

Hilt

Adding dependency

build.gradle (app)

```
...
apply plugin: 'kotlin-kapt'
apply plugin: 'dagger.hilt.android.plugin'

android {...}

dependencies {
    implementation "com.google.dagger:hilt-android:2.28-alpha"
    kapt "com.google.dagger:hilt-android-compiler:2.28-alpha"
}
```

Hilt

Enable Java 8

build.gradle (app)

```
android {
    ...
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Hilt

Application class

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

Hilt

Application class

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

Hilt

Inject into Activity

```
@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

    @Inject lateinit var myAwesomeHelper: MyAwesomeHelper
    @Inject lateinit var retrofitService: RetrofitService

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        myAwesomeHelper.doAwesomeThings()
        // retrofitService is ready to use here.
    }
}
```

Hilt

Inject into Activity

```
@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

    @Inject lateinit var myAwesomeHelper: MyAwesomeHelper
    @Inject lateinit var retrofitService: RetrofitService

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        myAwesomeHelper.doAwesomeThings()
        // retrofitService is ready to use here.
    }
}
```

Hilt

Inject into Activity

```
@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

    @Inject lateinit var myAwesomeHelper: MyAwesomeHelper
    @Inject lateinit var retrofitService: RetrofitService

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        myAwesomeHelper.doAwesomeThings()
        // retrofitService is ready to use here.
    }
}
```

Supported Android classes

- Application
- Activity
- Fragment
- View
- Service
- BroadcastReceiver

Hilt

Setting up module for third-party classes

```
@Module
@InstallIn(ApplicationComponent::class)
class NetworkModule {

    @Provides
    fun provideRetrofitService(): RetrofitService {
        ...
    }
}
```

Hilt

Setting up module for third-party classes

```
@Module
@InstallIn(ApplicationComponent::class)
class NetworkModule {

    @Provides
    fun provideRetrofitService(): RetrofitService {
        ...
    }
}
```

Supported Component classes

- ApplicationComponent
- ActivityRetainedComponent
- ActivityComponent
- FragmentComponent
- ViewComponent
- ViewWithFragmentComponent
- ServiceComponent

Hilt and Jetpack 

Hilt and Jetpack

Adding dependency

```
apply plugin: 'kotlin-kapt'

dependencies {
    ...

    implementation 'androidx.hilt:hilt-lifecycle-viewmodel:1.0.0-alpha01'
    implementation 'androidx.hilt:hilt-work:1.0.0-alpha01'
    kapt 'androidx.hilt:hilt-compiler:1.0.0-alpha01'

}
```

Hilt and Jetpack

Prepare ViewModel

```
class MyViewModel @ViewModelInject constructor(
        private val myAwesomeHelper: MyAwesomeHelper
): ViewModel() {

    fun doSomething() {
        myAwesomeHelper.doAwesomeThings()
    }

}
```

Hilt and Jetpack

Prepare ViewModel

```
class MyViewModel @ViewModelInject constructor(
        private val myAwesomeHelper: MyAwesomeHelper
): ViewModel() {

    fun doSomething() {
        myAwesomeHelper.doAwesomeThings()
    }

}
```

Hilt and Jetpack

Inject into Activity

```
@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

    private val viewModel by viewModels<MyViewModel>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        viewModel.doSomething()
    }
}
```

Hilt and Jetpack

Inject into Activity

```
@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

    private val viewModel by viewModels<MyViewModel>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        viewModel.doSomething()
    }
}
```

Hilt and Jetpack

Inject into Activity

```
@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

    private val viewModel by viewModels<MyViewModel>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        viewModel.doSomething()
    }
}
```

Reference links

- Dependency injection in Android (guide)
- Dependency injection with Hilt (guide)
- Hilt documentation (doc)
- Dependency Injection on Android with Hilt (article)
- Using Hilt in your Android app (codelab)
- Migrating your Dagger app to Hilt (codelab)

Question?

Thank you.