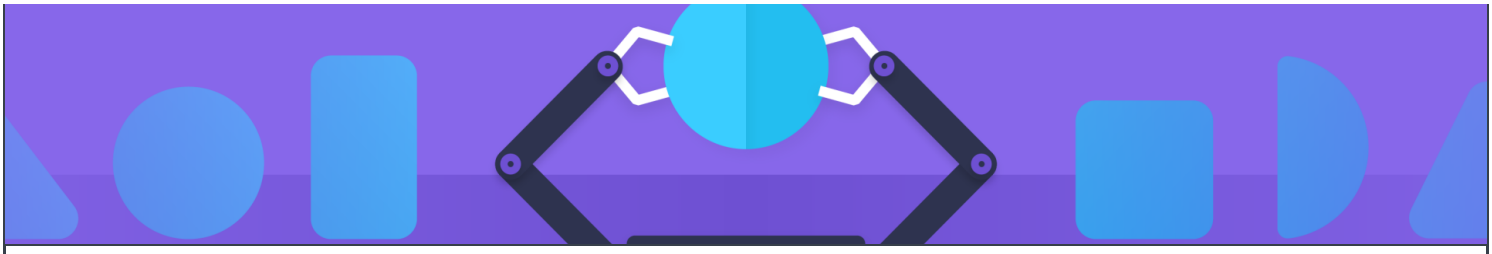

JUNE 23RD, 2020 | 2 MINUTE READ

Introducing Anvil

Make dependency injection with Dagger 2 easier

Our Point of Sale Android codebase is a large modularized repository consisting of thousands of Gradle modules and hundreds of applications, each with its own unique Dagger 2 dependency graph. When creating a new application, one of the biggest bottlenecks we've seen in developer productivity was setting up Dagger dependencies. It's a tedious process and involves many steps:

1) Try to build the application and find out from the build failure which dependencies are missing.
2) Add a Gradle dependency to the modules containing the Dagger modules with the missing bindings.
3) Sync the IDE.
4) Add the Dagger modules to the Dagger components.
5) Repeat.

A similar process is required when creating a new Dagger module that should be included in applications.

We didn't want to question Dagger as the status quo, because the compile-time dependency graph validation for that many applications is invaluable. Instead, we wanted to make working with Dagger easier and the cycle from above shorter.

A small example could be as simple as this:

```
@Module
@ContributesTo(AppScope::class)
class DaggerModule { .. }

@MergeComponent(AppScope::class)
interface AppComponent
```

Instead of adding `DaggerModule` to `AppComponent` directly, we tell Anvil to make this connection for us. The code you'd write manually without Anvil looks like this:

```
@Module
class DaggerModule { .. }

@Component(modules = [DaggerModule::class])
interface AppComponent
```

With this tiny example the gains especially for developer time aren't obvious. However, time savings become more clear as soon as the Dagger module is shared with more application targets. Instead of adding the module to all application components, it's only necessary to declare once in which scope it should be included. In the cycle above we avoid steps 3 and 4, the IDE sync and adding Dagger modules to Dagger components. They are redundant with Anvil.

Anvil brings other advantages, e.g. implicit scopes in modules become explicit, the Dagger dependency graph is aligned with the Gradle build graph and we can easily decompose now redundant large composite Dagger modules.

practices can be found on the Github page of the project.

What about Hilt, Google's new dependency injection framework for Android? It provides a similar feature with @InstallIn, and if you're using Hilt you don't need Anvil. But Hilt has a lot of features we don't need, and imposes restrictions we can't live with — you might find the same. Details are available on Github.

We're looking forward to hearing your feedback.

AUTHORED BY

Ralf Wondratschek

TAGS

Dagger 2 Open Source Mobile Development

Discuss on Twitter

Join our Slack Channel

APIs

Engineering

Data Science

View More Articles ›