How does Transfuse compare with Dagger?

Asked 11 years ago Modified 8 years, 5 months ago

Viewed 837 times Part of Mobile Development Collective

6

I'm trying to decide whether to use Transfuse or Dagger for Android dependency injection. I've never used Transfuse, and have basic knowledge of Dagger. Thanks much.

MD

android dependency-injection comparison dagger transfuse

Share

Improve this question

Follow

asked Nov 30, 2013 at 23:10

Julian A.

11.5k ● 16 ● 73 ● 109

1 Answer

Sorted by: Highest score (default)

12

To start, I am the primary author of Transfuse thus this answer may be a bit slanted in that direction.

Both Transfuse and Dagger handle Dependency Injection / Inversion of Control for Android in similar ways. Both

use Annotation Processing at Compile time via JSR269 to generate code the supports the DI/IOC functionality. This allows them to avoid the costly runtime reflection-based analysis typically associated with DI containers found in non-Android Java. Without going into the specifics, Dagger and Transfuse do approach code generation in significantly different ways, which is reflected in the features of the libraries. Also, Transfuse and Dagger both use the common JSR330 annotations (@Inject, Provider, etc). This means they both follow a Guice-style injection scheme.

Here's how you create an object graph in Dagger:

```
public class DaggerActivity extends Activity {

    @Inject Example example;

    @Override protected void onCreate(Bundle savedInst
        super.onCreate(savedInstanceState);
        ObjectGraph.create().inject(this);
        //do something else...
    }
}
```

The equivalent code in Transfuse uses its @Factory functionality:

```
@Factory
public interface Injector {
    Example get();
}

public class TransfuseActivity extends Activity {
```

```
    Example example;

    @Override protected void onCreate(Bundle savedInst
        super.onCreate(savedInstanceState);
        example = Factories.get(Injector.class).get();
        //do something else...
    }
}
```

Transfuse is meant to be used in the following way, however, utilizing POJO components, lifecycle events, etc:

```
@Activity
public class TransfuseActivity{

    @Inject Example example;

    @OnCreate public void doSomethingElse(){
        //do something else...
    }
}
```

Here's some little differences in the DI engine in Transfuse and Dagger:

1. Transfuse supports (as well as it can) cyclic dependencies, Dagger purposefully throws an exception in this case.
2. Transfuse satisfies JSR330, Dagger specifically does not. The Dagger developers wanted to err on the side of simplicity, avoiding method injection allowed them to avoid a handful of confusing cases (link).

3. Dagger has a reflection based engine for cases where code was not generated. Transfuse does not and requires code to be generated (annotation processor to run) in order to work.
4. Transfuse will inject into private fields, constructors, methods (not necessarily recommended because of the reflection overhead). Dagger throws an exception in this case.
5. Dagger uses Modules in a very direct way, mirroring the capabilities of Guice. Each time you create an object graph, you are given the option to configure it with a Module class, ie: `ObjectGraph.create(new DripCoffeeModule())`. Transfuse's configuration module is a bit different as it is incorporated into the application at compile time. Each Module in Transfuse is global to the project (this may change in future versions of Transfuse, but it has not been an issue for the use of Transfuse yet).
6. Singletons in Dagger are per-object-graph where Singletons in Transfuse are global to the application.

The big difference between Dagger and Transfuse is that Dagger focused on being a simple Dependency Injection library, while Transfuse's focus is to "make Android a better API using performance sensitive techniques"

Transfuse supports these capabilities as well as DI:

1. POJO Components
2. Manifest Management

3. Roboguice/Butterknife style injections
4. Lightweight event system (@Observes, @OnCreate, etc)
5. AOP

I'd recommend that if you're interested, give Transfuse a try. Personally, I'd love to hear about your experience contrasting it with Dagger. We have a mailing list where you can share with the community and pretty through documentation on the website.

Share Improve this answer Follow

edited Jun 26, 2016 at 20:26

answered Dec 1, 2013 at 15:49

John Ericksen
11.1k ● 4 ● 47 ● 75

@johncarl It's FANTASTIC library! But I wonder why is it not so widely used as AndroidAnnotations and Dagger, while being much more complete and robust? Can I rely on it in my application and be sure you won't abandon it at some moment in the future? – afrish Jul 27, 2014 at 19:52

Answered nucleo on Google Groups: groups.google.com/forum/#!topic/transfuse/qn3DFUMxzas – John Ericksen Aug 5, 2014 at 14:28

@nucleo as it said, Dagger is a just nice simple DI without complicating with other features or requirement. Me myself find Transfuse more useful but using Dagger in most of my App due to simplicity of implementation. – Azri Jamil Dec 31, 2014 at 14:56