dagger2 使用教程终章

九风特 关注
2020.09.21 10:06:32 字数 630 阅读 470

终章引言

关于dagger2的学习，差不多到此为止了，不过我突然发现我在第三节提过@BindsInstance，本节补上，这东西还是挺重要的

@BindsInstance使用说明

这个注解将组件生成器上的方法或组件工厂上的参数标记为实例并绑定到组件内的某个键。。听不明白没关系，来看实例就啥都明白了，还是改造之前的类，就拿现在还比较简单的Computer类开刀（注意本教程，必须从头学，不然类名看不懂），在额外加一个价格的参数

```
class Computer(private val name:String, private val price:Int){
    override fun toString() = "Computer:$name, Price:$price"
}
```

我们在之前的provider这样提供的Computer实例

```
@Module
class ComputerModule{
    @Provides
    fun getComputer()=Computer("戴尔", 100)
}
```

这肯定没错，但显然不合适，怎么可能所有的电脑都是“戴尔”，价格都是“100”呢，这俩参数显然应该由用户在创建实例的时候自行决定(关于用@Name实现这种意图的方式，可看第三节)那咋整呢，那就是得让Dagger知道，创造computer时需要用户制定俩参数，怎么告诉Dagger这件事呢?

先来改造Computer类

```
class Computer @Inject constructor(private val name:String, private val price:Int){
    override fun toString() = "Computer:$name, Price:$price"
}
```

去掉原来的ComputerModule，改造Component

```
@Component
interface ComputerComponent{
    fun makeComputer():Computer
    @Component.Builder
    interface Builder{
        @BindsInstance fun name(name:String):Builder
        @BindsInstance fun price(price:Int):Builder
        fun build():ComputerComponent
    }
    fun makeVirtualComponent():SubComponentVirtualPrinter//此接口为前几节遗留
}
```

这个代码的@Component.Builder意思就是在重构Component的Builder类，通过@BindsInstance最终给Component增加了两个字段name和price, 构建Computer实例时会使用他们（根据类型dagger很容易知道哪个对哪个)。 在改写下UI代码

```
        val cpuCom = DaggerComputerComponent.builder()
            .name("戴尔")
            .price(1200)
            .build()
```

最终一切正常，这时你如果给Computer再增加个参数叫productName会如何?

```
class Computer @Inject constructor(private val name:String,
                                   private val price:Int,
                                   private val productName:String){
    override fun toString() = "Computer:$name, Price:$price, ProductName:$productName"
}

@Component
interface ComputerComponent{
    fun makeComputer():Computer
    @Component.Builder
    interface Builder{
        @BindsInstance fun price(price:Int):Builder
        @BindsInstance fun name(name:String):Builder
        @BindsInstance fun pName(str:String):Builder//为了避免造成名字必须一致的误解，此处名
        fun build():ComputerComponent
    }
    fun makeVirtualComponent():SubComponentVirtualPrinter//此接口为前几节遗留
}

        val cpuCom = DaggerComputerComponent.builder()
            .name("戴尔")
            .price(1200)
            .pName("Dell_4700")
            .build()
```

看起来没啥毛病，但现在dagger已经无法知道新注入的3个字段怎么使用了，我们必须明确告诉dagger，这可以使用我们之前提过的@Named 或者自定义修饰符,稍微改下上面代码即可

```
class Computer @Inject constructor(@Named("arg1") private val name:String,
                                   @Named("arg2") private val price:Int,
                                   @Named("arg3") private val productName:String){
    override fun toString() = "Computer:$name, Price:$price, ProductName:$productName"
}
@Component
interface ComputerComponent{
    fun makeComputer():Computer
    @Component.Builder
    interface Builder{
        @BindsInstance fun price(@Named("arg2")price:Int):Builder
        @BindsInstance fun name(@Named("arg1")name:String):Builder
        @BindsInstance fun pName(@Named("arg3")str:String):Builder//为了避免造成名字必须一
        fun build():ComputerComponent
    }
    fun makeVirtualComponent():SubComponentVirtualPrinter//此接口为前几节遗留
}
```

现在一切okay了，个人建议如果参数比较多，不管是否有重复的类型都加上合适的修饰吧。
还有一些不太常用的注解，官方文档都没说，本文也就不解释了，真要用可以看看资料
@IntoMap
@IntoSet

结语:Dagger是一个用于解耦的工具，如果你的项目有一定的复杂度，就应该考虑使用它，这东西使用成本确实比较高。如果是简单的项目就算了吧。