Mateusz Herych

Android Developer @ Base CRM

Co-organizer @ GDG Kraków

partition @ GitHub

Inject with Dagger

DevFest 2013, Berlin

Problem?

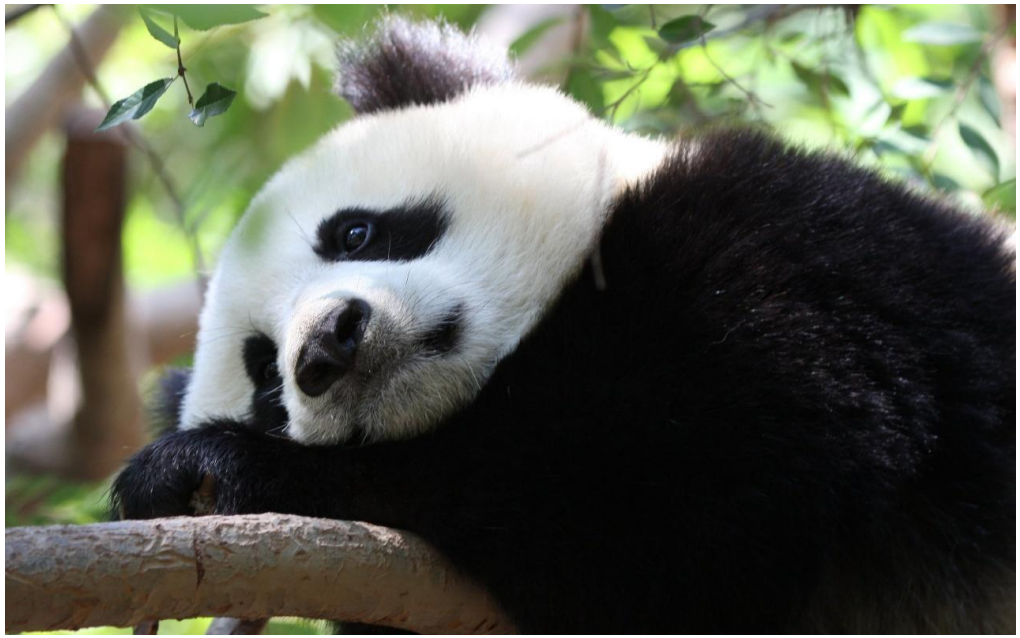
```
public class Api {
  private Http http = new Http();

  // Abstraction over REST methods.
}
```

```
@RunWith(RobolectricTestRunner.class)
public class MyTestSuite {
  @Test public void shouldMockOutHttp() {
   }
}
```

Unit Tests

- Fast
- Simple
- Unit!

```
public class MyTestSuite {
  @Test public void shouldMockOutHttp() {
```



```
  }
}
```

Solution?

```
public class Api {
  public Http http;

  // Abstraction over api methods, uses http.
}
```

Solution?

```
public class Api {
  private Http http;

  public void setHttp(Http http) {
    this.http = http;
  }
}
```

Solution?

```
final Api api = new Api();
api.apiMethod(); // NPE
```

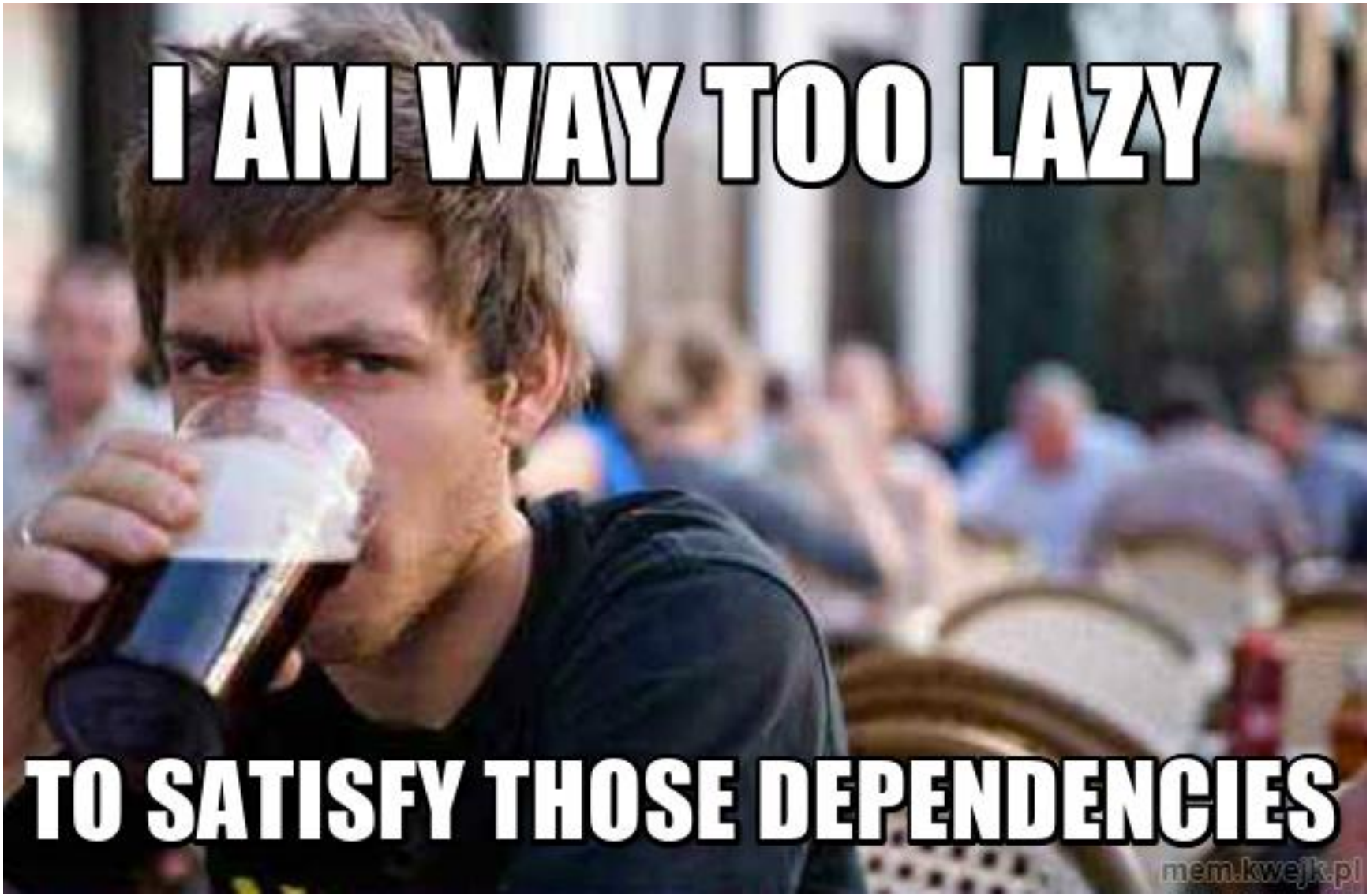
Solution?

```
public class Api {

  private final Http http;

  public class Api(Http http) {
    this.http = http;
  }
}
new Api(new Http()); --- ?
```

Ok, but...

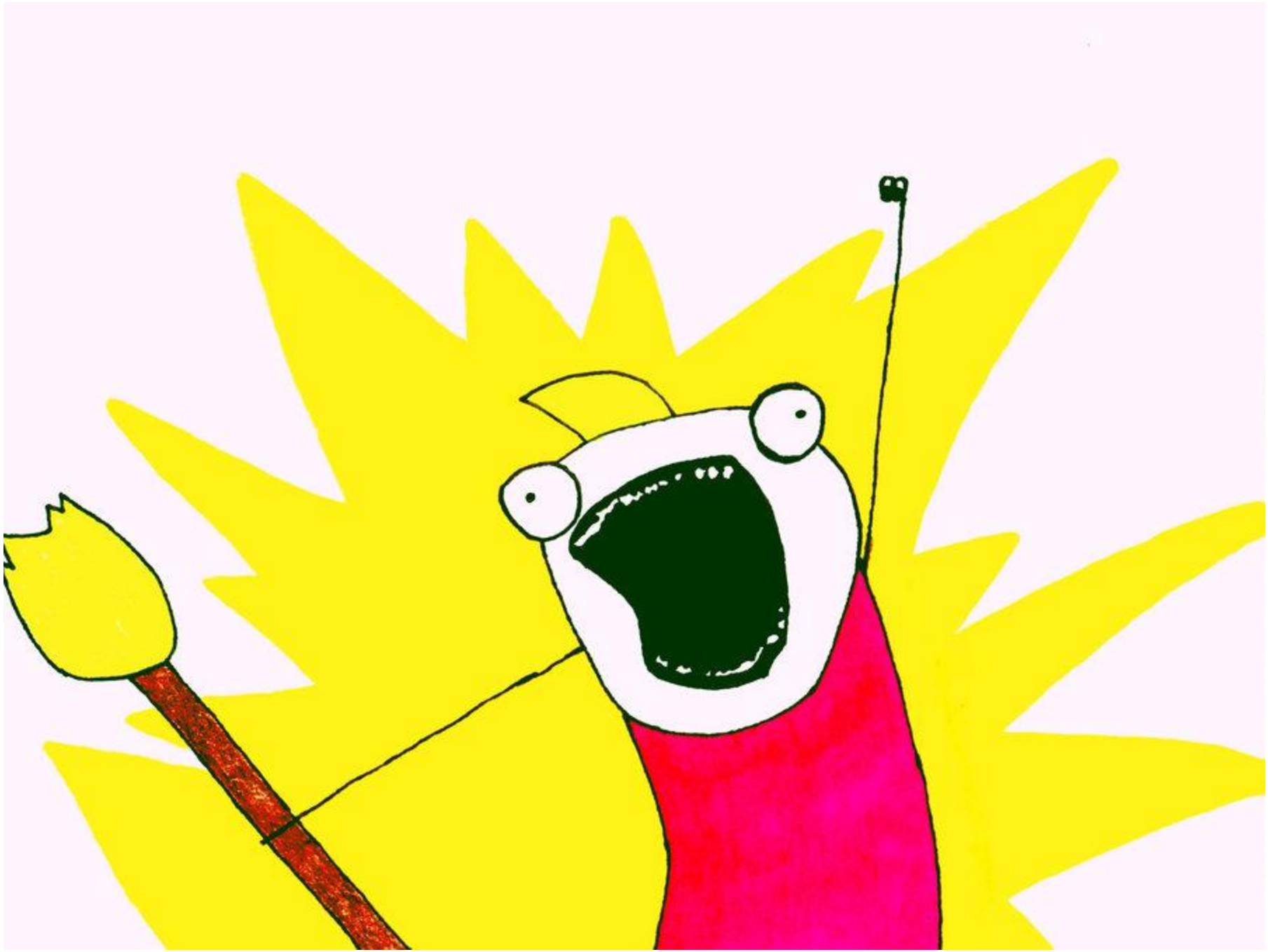
```
public Http(Dependency1 bleh);
public Dependency1(Dependency2 blah,
  Logger logger);
public Dependency2(JużŁapiecie lol);
```

Ok, but...



Ok, but...

DRY.



Ok, but...

Ok, but...

Dependency Injection FTW!

```
public class Api {
  private final Http http;

  @Inject
  public Api(Http http) {
    this.http = http;
  }
}
```

Android

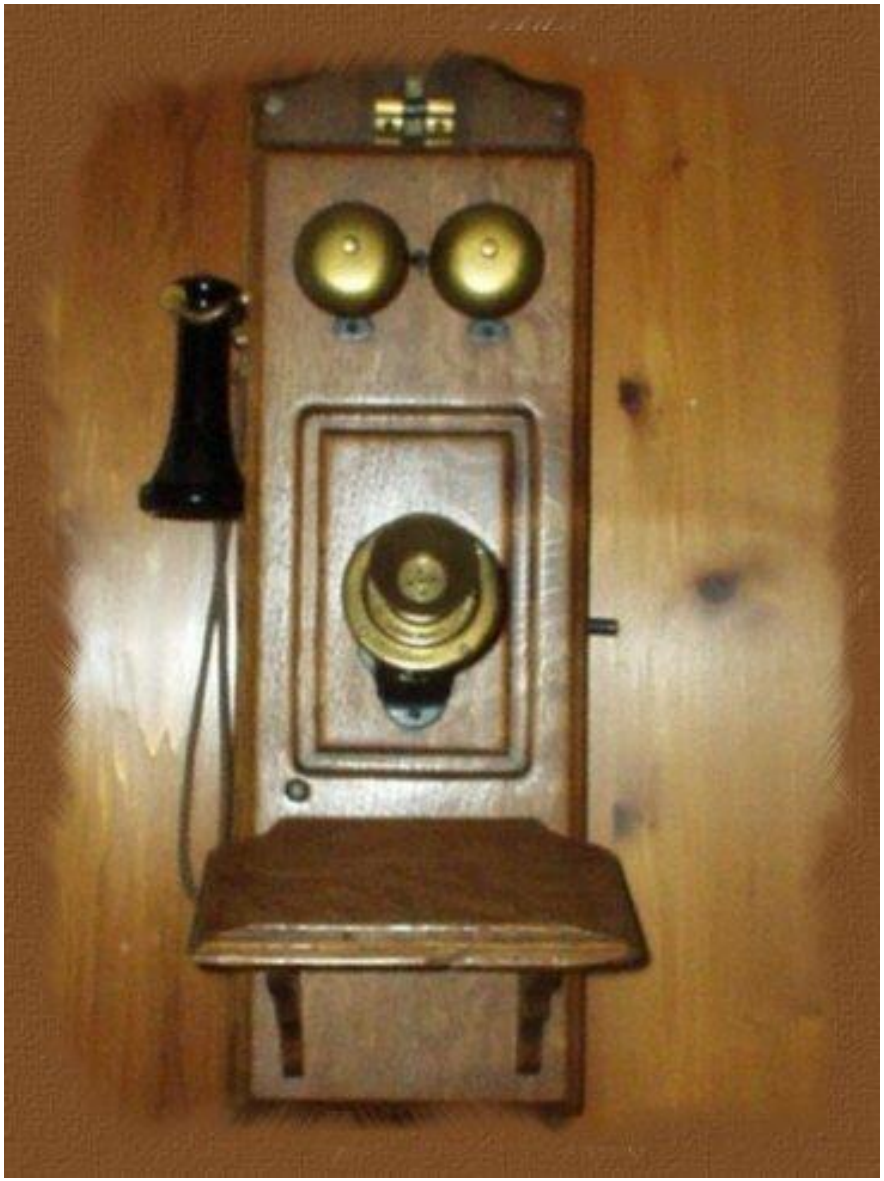
Nothing’s new

RoboGuice

- Guice-based
- Android (@Inject LayoutInflater, etc.)
- @InjectView
- Reflection

“If You're Programming A Cell Phone Like A Server You're Doing It Wrong”

-- highscalability.com




htc
8 37
Phone

RoboSplashActivity

Dagger

- Light framework for Java and Android
- Codegen
- Compile-time validation
- Almost all interesting features of DI. Almost.

Hello World

```
public class SimplestInjection {

  @Inject
  HappyGreeter greeter;

  public SimplestInjection() {
    ObjectGraph.create(MyModule.class).inject(this);
    System.out.println(greeter.greet("Janusz"));
  }

  @Module(injects = SimplestInjection.class)
  static class MyModule {

  }

}
```

```
public class MultipleImplementationsInterfaceInjection {

  @Inject
  Greeter iAmHappy;

  public MultipleImplementationsInterfaceInjection() {
    ObjectGraph.create(HappyModule.class).inject(this);
  }

  @Module(injects = MultipleImplementationsInterfaceInjection.class)
  static class HappyModule {

    @Provides
    public Greeter provideHappyGreeter() {
      return new HappyGreeter();
    }
  }
}
```

```
@Module(injects = MultipleImplementationsInterfaceInjection.class)
static class HappyModule {

  @Provides
  public Greeter provideGreeter(HappyGreeter happyGreeter) {
    return happyGreeter;
  }
}
```

```
public class SingletonInjection {

  @Inject
  SingletonGreeter iAmASingleton;

  @Inject
  Greeter iAmASingletonToo;

  public SingletonInjection() {
    ObjectGraph.create(SingletonModule.class).inject(this);
  }

  @Module(injects = SingletonInjection.class)
  static class SingletonModule {

    @Provides
    @Singleton
    public Greeter provideSingleton() {
      return new HappyGreeter();
    }
  }
}
```

Constructor!

```
public class GreeterCensoringProxy {

  private Greeter greeter;

  @Inject
  public GreeterCensoringProxy(HappyGreeter greeter) {
    this.greeter = greeter;
  }

  public String greet(String name) {
    return "Censored.";
  }
}
```

Not everyone is happy

```
@Inject @Named("sad")
Greeter sad;

@Inject @Named("happy")
Greeter happy;

public NamedInjection() {
  ObjectGraph.create(NamedModule.class).inject(this);
  System.out.println(sad.greet("Janusz"));
  System.out.println(happy.greet("Janusz"));
}
```

But everyone deserves a name!

```
@Module(injects = NamedInjection.class)
static class NamedModule {

  @Provides @Named("happy")
  public Greeter provideHappyGreeter() {
    return new HappyGreeter();
  }

  @Provides @Named("sad")
  public Greeter provideSadGreeter() {
    return new SadGreeter();
  }
}
```

```
@Qualifier
@Retention(RetentionPolicy.RUNTIME)
public @interface Activity {

}
```

```
public class QualifiedInjection {

  @Inject @Activity
  Greeter sad;

  @Inject @Application
  Greeter happy;

}
```

Validation

No injectable members on com.github.partition.common.SadGreeter. Do you want to add an injectable constructor? required by com.github.partition.ScopedInjection.First for com.github.partition.ScopedInjection.AdditionalModule

How?

JSR269

How?

“Pluggable annotation processing API”

Now I'll tell what doesn't work.

```
@Inject
private MyDependency dependency;
```

```
@Inject
public void setDependency(Dependency d) {
  // stuff.
}
```

Proguard

@Keep for the rescue!

```
@Keep
public class MyDependency {
  // Dagger is manipulating with me, so please
  // leave me alone, proguard!
}
```

How to use it?

dagger.jar
dagger-compiler.jar

(sure, you'll find it in maven central!)


SPAM®
with Real HORMEL® BACON
U.S. INSPECTED AND PASSED BY DEPARTMENT OF AGRICULTURE
NET WT 12 OZ (340g)
Serving Suggestion


GDG Kraków

KrakDroid, 7.12.2013, Kraków