GETTING STARTED WITH

DEPENDENCY INJECTION AND GUICE

Justin Lee / @evanchooly

Member of Technical Staff @10gen

PEDIGREE

- JVM mongodb drivers team at 10Gen (morphia)
- JSR 356 Expert Group Member
- Grizzly Websockets
- More side projects than time
  - @ github
- but enough of that

SO. INJECTION.

Also known as Inversion of Control

though some make a broader distinction

Normally you give the class what it needs

You have to know how to construct the class

And all of its dependencies

which probably have their own dependencies

that have dependencies

that have dependencies

that have dependencies

that have dependencies

that have dependencies

SO. WHAT CAN WE DO ABOUT THAT?

We can inject our dependencies!

Right. But what does that even mean?

Well, it can mean a few things

In short, the class declares its dependencies somehow

XML (Spring)

Annotations (Guice and CDI)

We're gonna focus on guice obviously

DIFFERENT KINDS OF INJECTION

Field

Constructor

com.google.inject.Inject

(javax.inject.Inject -- CDI)

FIELD INJECTION

```
public class Needy {
    @Inject
    private MailService mailer;
    @Inject
    private Invoice invoice;
}
```

CONSTRUCTOR INJECTION

```
public class Needy {
    private MailService mailer;
    private Invoicer invoicer;

    @Inject
    public Needy(MailService mailer, Invoicer invoicer) {
        this.mailer = mailer;
        this.invoice = invoice;
    }
}
```

DEFINING THE DEPENDENCIES

```
public class GuiceModule extends AbstractModule {
  @Override
  protected void configure() {
    bind(BlahBlah.class).to( BlahBlahImpl.class );
  }

  @Provides
  public MailService create() {
    return new MailService("mail.example.com", 1500,
        "myusername", "mypassword");
  }
}
```

PUTTING IT ALL TOGETHER

```
Injector injector = Guice.createInjector(new GuiceModule());
Needy needy = injector.getInstance(Needy.class);
```

ENOUGH SLIDEWARE. LET’S SEE SOME CODE.