DODO ENGINEERING

The art of JUnit Extensions

#2

Дмитрий Тучс

Heisenbug 2024 Spring online

Head of QA @ Dodo Engineering

dtuchs (и https://t.me/likeaduck)

Преподаю в QA.GURU

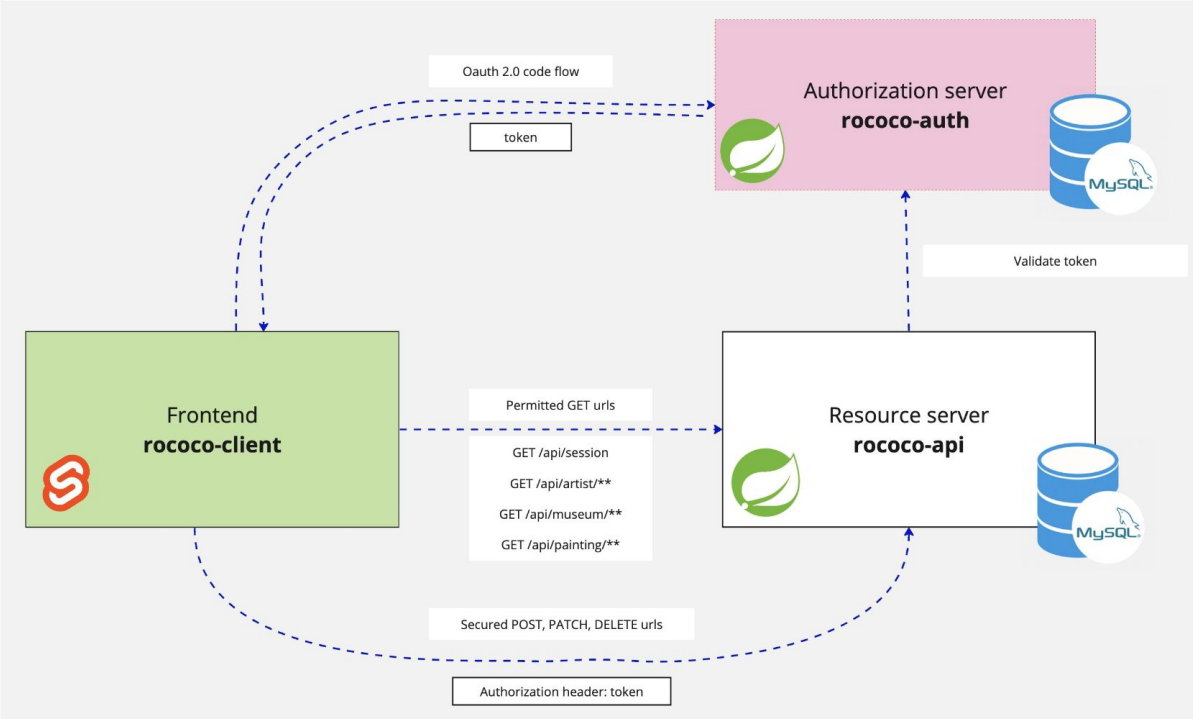
Опыт в разработке, аналитике, управлении проектами более 14 лет

Для кого?

- Для всех, кто использует JUnit
- Для тех, кому интересно вместе со мной доделать *элегантные тесты*
- Первую часть можно посмотреть и после :)

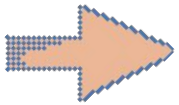
Первая часть – где?

Первая часть – что сделали?



https://github.com/dtuchs/heisenbug-2023-autumn

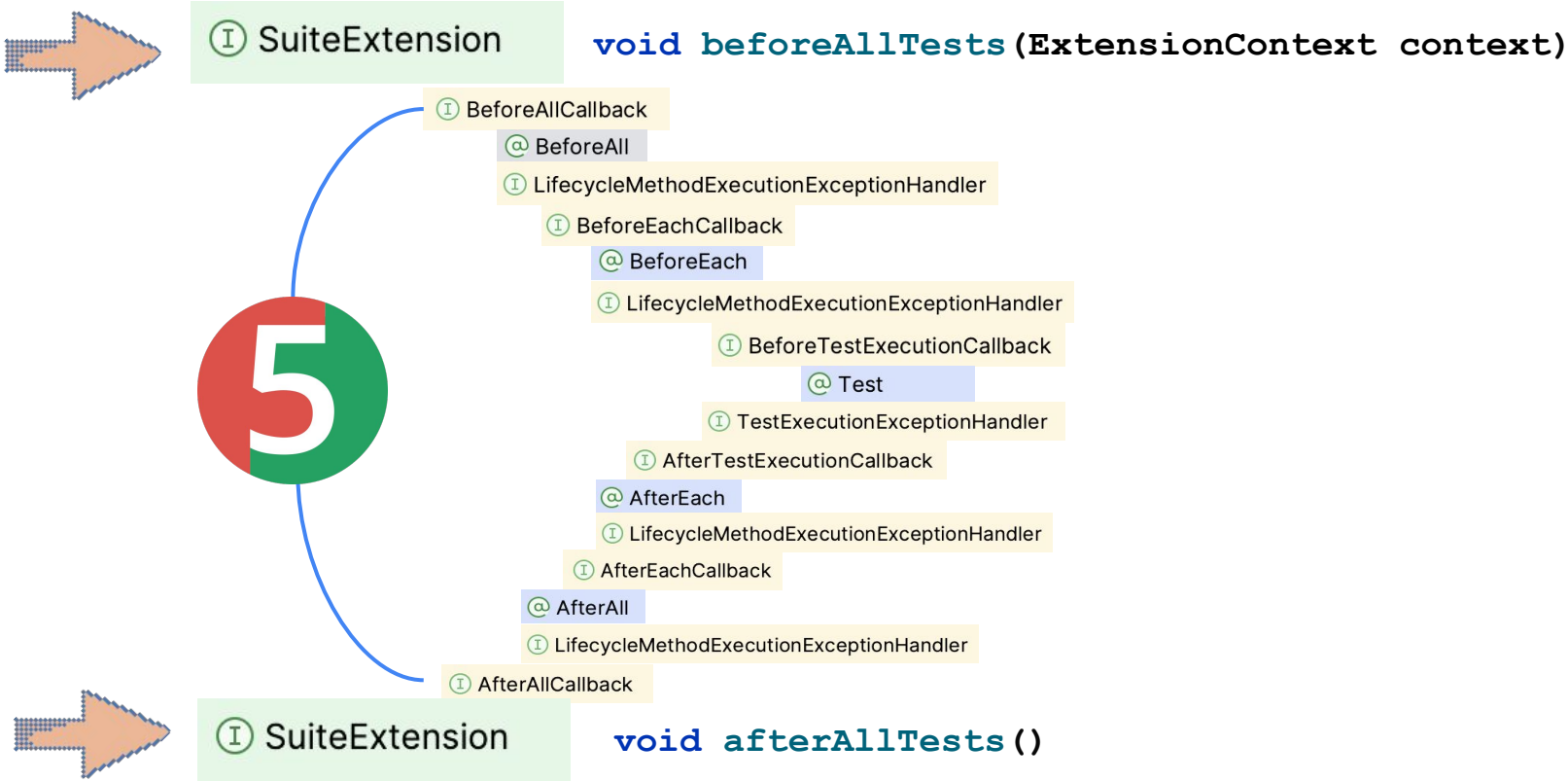
Первая часть: OAuth API login



```
@ApiLogin(username = "dima", password = "12345")
@Test
void usernameShouldBeVisibleInProfile() {
 $("[data-testid='avatar']").click();
 $("h4.text-center").should(text("dima"));
}
```

https://github.com/dtuchs/heisenbug-2023-autumn

Первая часть: создали свой интерфейс

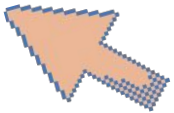


Первая часть: создали свой интерфейс

```
class PreconditionsExtension implements SuiteExtension{

  @Override
  public void beforeAllTests(ExtensionContext context)
  {
      DELETE FROM `rococo-api`.`user`;

      INSERT INTO `rococo-api`.`user` (`id`, username, firstname, lastname, avatar)
      VALUES (0x881177007F564347B2619F966690BB99, 'petr', null, null, null);
      INSERT INTO `rococo-api`.`user` (`id`, username, firstname, lastname, avatar)
      VALUES (0x8A68F43027A04533BADA95CB0103D73D, 'valentin', null, null, null);
  }
```



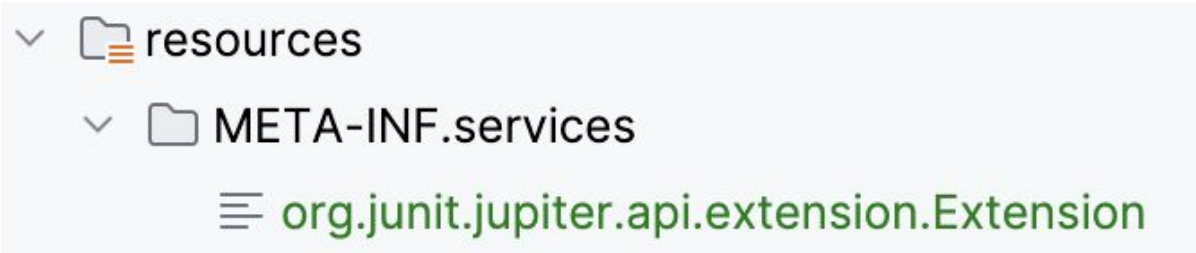
https://github.com/dtuchs/heisenbug-2023-autumn

Первая часть: 2/3 способа регистрации

Declarative Extension Registration

```
@ExtendWith({ContextHolderExtension.class,
ApiLoginExtension.class})
public class WebTest
```

Automatic Extension Registration



https://github.com/dtuchs/heisenbug-2023-autumn

Первая часть: и один PR за кадром

```
32      @POST("oauth2/token")
33  +   @FormUrlEncoded
34      Call<JsonNode> token(
35          @Header("Authorization") String basicAuthorization,
    -       @Query("client_id") String clientId,
    -       @Query(value = "redirect_uri", encoded = true) String redirectUri,
    -       @Query("grant_type") String grantType,
    -       @Query("code") String code,
    -       @Query("code_verifier") String codeVerifier
36  +       @Field("client_id") String clientId,
37  +       @Field(value = "redirect_uri", encoded = true) String redirectUri,
38  +       @Field("grant_type") String grantType,
39  +       @Field("code") String code,
40  +       @Field("code_verifier") String codeVerifier
41      );
```

https://github.com/dtuchs/heisenbug-2023-autumn/pull/1/files

Первая часть: и один PR за кадром

LikeaDuck🦆
#Java #Spring

Об обновлении зависимостей.

Обновил тут в одном из своих Spring пет-проектов одну зависимость.
Было 1.1.2, стало - 1.2.1. Поменял местами 2 цифры так сказать, что могло бы пойти не так?

У меня напрочь отвалился Oauth 2.0 code flow.

https://t.me/likeaduck/61

Прежде чем **продолжим** писать код

- А с браузером, с REST тестами? И с Allure?
- А если тест *вдруг* найдет баг?
- А что с *локальными* тестовыми данными?

https://github.com/dtuchs/heisenbug-2023-autumn

И пара прикладных вопросов

- А третий способ регистрации Extension?
- А параллельно нормально будет работать?
- И немного порефакторить?

https://github.com/dtuchs/heisenbug-2023-autumn

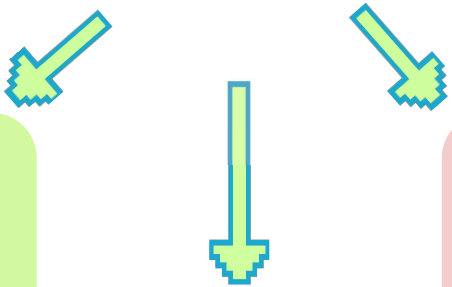
И наконец

Изящные тесты с нуля за 2 часа – это реально?

https://github.com/dtuchs/heisenbug-2023-autumn

Declarative Extension Registration

```
@ExtendWith({ContextHolderExtension.class,
ApiLoginExtension.class})
```



Стандартное решение

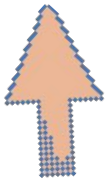
Потокобезопасность только **в ваших руках**

Extension instance - **singleton**

Declarative Extension Registration

```
public class ApiLoginExtension implements
BeforeEachCallback, AfterEachCallback {

 private final AuthClient authClient = new AuthClient();
```

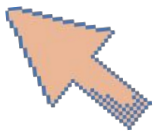


API Client –
фактически static

Declarative Extension Registration

```
public class DatabaseCreateUserExtension {

    private final UserRepository userRepository = new
UserRepositoryHibernate();
```



Объекты, работающие с БД - *фактически static*

Declarative Extension Registration

Thread-safe dependencies

```
class OkHttpClient

class HttpClient

class MysqlDataSource (DS)

class SessionFactoryImpl (EMF)
```

Declarative Extension Registration

Thread-safe design

```
class OkHttpClient

class HttpClient
```

```
public enum ThreadLocalCookieStore implements CookieStore {
  INSTANCE;

  private final ThreadLocal<CookieStore> store = ThreadLocal.withInitial(
      () -> new CookieManager(null, ACCEPT_ALL).getCookieStore()
  );
```

Declarative Extension Registration

Thread-safe design

class MysqlDataSource

```
public Optional<UserEntity> findById(UUID id) {
    try (Connection conn = dataSource.getConnection();
```

Declarative Extension Registration

Thread-safe design

```
class SessionFactoryImpl
```

```
private final EntityManager em =
        new TransactionalEntityManager(
                new ThreadSafeEntityManager(
                    emf.createEntityManager()
                )
        );
```

Programmatic Extension Registration

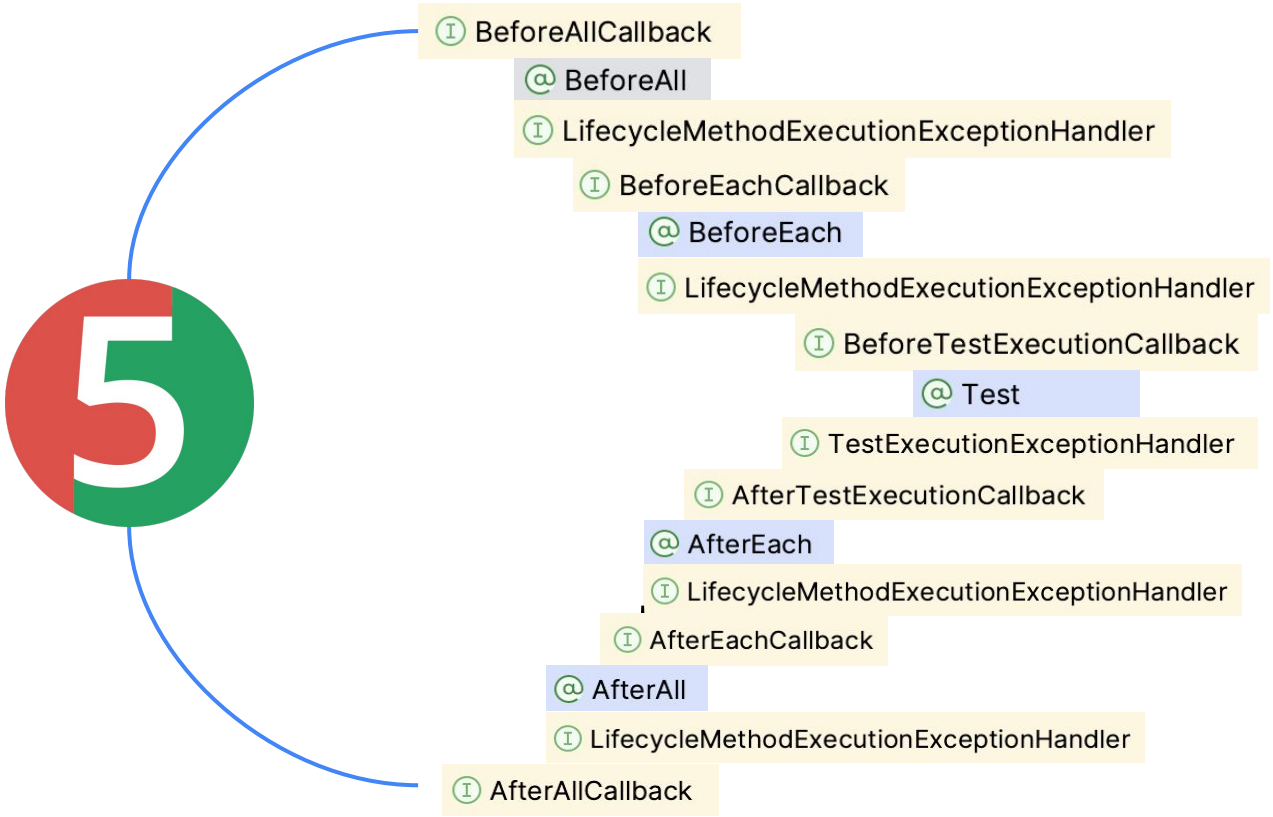
```
@RegisterExtension
private static final ApiLoginExtension loginExtension =
    new ApiLoginExtension(false);
```

Наделяем один Extension разным поведением

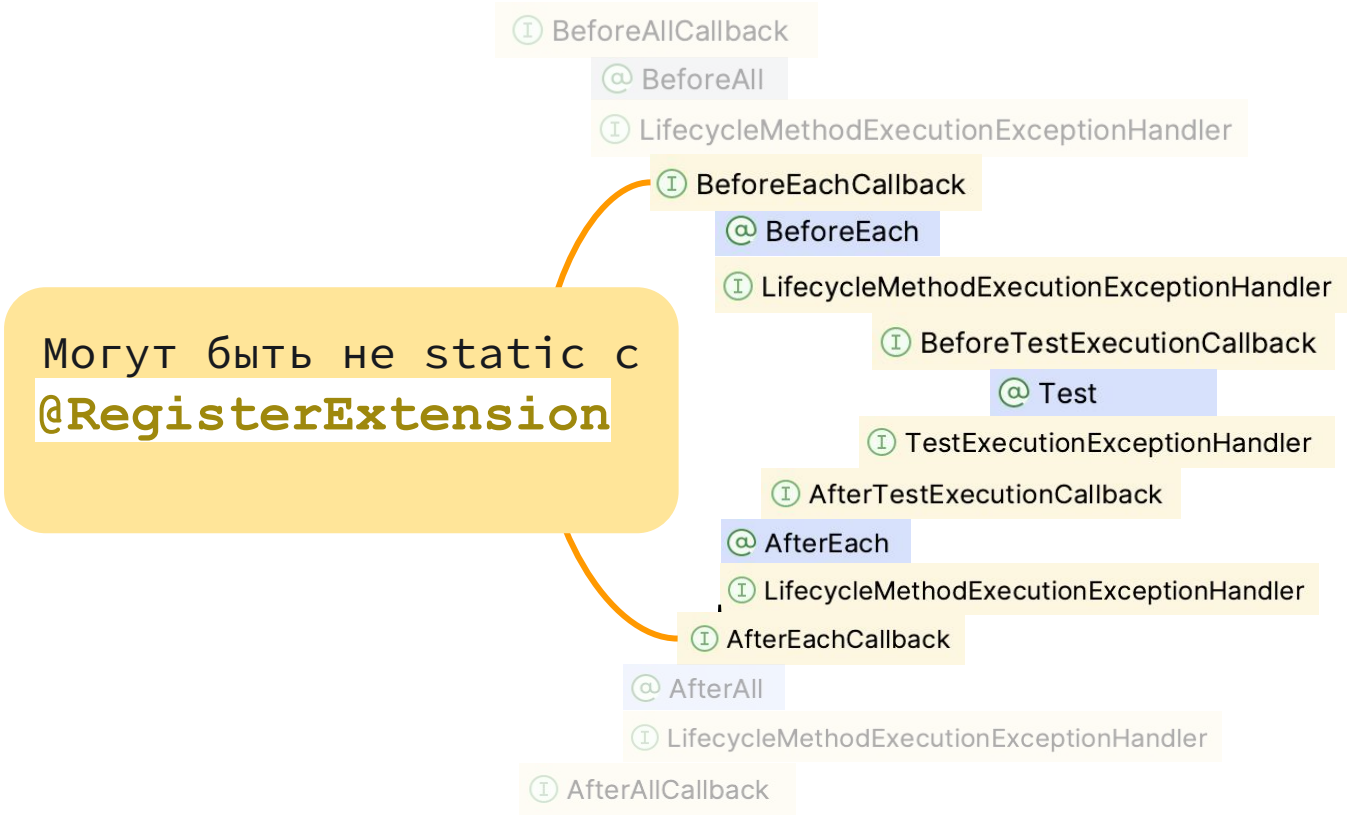
Может **не быть singleton**, если поле не static

но, только начиная с BeforeEach и заканчивая AfterEach

Programmatic Extension Registration



Programmatic Extension Registration



Global Extension Registration

org.junit.jupiter.api.extension.Extension

```
1  guru.qa.rococo.jupiter.PreconditionsExtension
```

Когда нужен **строго для всех** тестов

В остальном ведет себя как @ExtendWith **с высшим приоритетом**

Global Extension Registration

Чистим ресурсы

```
public class JpaExtension implements SuiteExtension {

    @Override
    public void afterAllTests() {
        EmfContext.storedEmf()
                .forEach(EntityManagerFactory::close);
    }
}
```

Global Extension Registration

Переделываем свой Allure

```
public class ServerLogsExtension implements SuiteExtension {

   private final String allurePath = "./proj/build/allure-results";

   @Override
   public void afterAllTests() {
     try (Stream<Path> paths =Files.walk(Path.of(allurePath))) {
           paths.filter(Files::isRegularFile)
                .filter(p -> p.toString().endsWith("-result.json"))
                   ...
```

Global Extension Registration

Отправляем отчеты

```
public class SendResultExtension implements SuiteExtension {

    private final String allurePath = "./proj/build/allure-results";

    @Override
    public void afterAllTests() {
        …
        allureApiClient.createProject(projectId);
        allureApiClient.sendResultsToAllure(
                projectId, new AllureResults(filesToSend)
        );
```

Вернемся к Rococo:

Demo 2: infra

https://github.com/dtuchs/heisenbug-2023-autumn

- **А с браузером, с REST тестами**? И с Allure?
- А если тест *вдруг* найдет баг?
- А что с *локальными* тестовыми данными?
- **А третий способ регистрации Extension?**
- **А параллельно нормально будет работать?**
- И немного порефакторить?

Почему Converter не Extension?

LikeaDuck🦆
#JUnit5 #Java
ArgumentConverter - Why you are not Extension? (

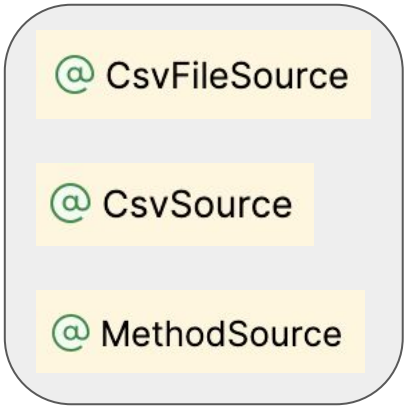
Я не понимаю, почему он не является Extension-ом.
Не я один - есть issue в статусе waiting-for-interest.

Ведет он себя очень похоже на любой другой Extension;
Подключаем через `@ConvertWith` вместо `@ExtendWith`.

https://github.com/junit-team/junit5/issues/853

Напишем отдельный Extension

Напишем отдельный Extension




```
@ArgumentsSource(
? extends AnnotationBasedArgumentsProvider
)
```

```
accept(A annotation)
```

```
provideArguments(context)
```

```
Stream<? extends Arguments>
```

```
skip(invocationIndex - 1).findFirst()
```

```
String allureId = (String)arg.get()[0]
```

Еще один путь

```
@BeforeEach
void doSmth(@AllureParamId String allureId, String painting) {

}

@CsvSource({
    "123, Утро в сосновом лесу",
    "124, Над вечным покоем"
})
@ParameterizedTest
void test(@AllureParamId String allureId, String painting) {}
```

https://github.com/junit-team/junit5/issues/944

- **А с браузером, с REST тестами? И с Allure?**
- **А если тест *вдруг* найдет баг?**
- А что с *локальными* тестовыми данными?
- **А третий способ регистрации Extension?**
- **А параллельно нормально будет работать?**
- И немного порефакторить?

Вернемся к Rococo:

Demo 3: preconditions

https://github.com/dtuchs/heisenbug-2023-autumn

DSL тестовых данных

```
data {
    startFrom("08:00")
    subjects("Russian",
            "Literature",
            "Algebra",
            "Geometry")
    student {
        name = "Ivanov"
        subjectIndexes(0, 2)
    }
    student {
        name = "Petrov"
        subjectIndexes(1, 3)
    }
```

DSL тестовых данных

ДОКЛАД Kotlin 24.04 / 19:15 – 20:00 (UTC+5)

One source to rule them all: Kotlin DSL как единый источник правды для решения многих задач

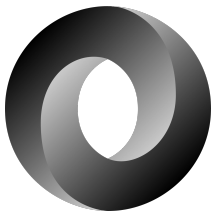
Зал 1 RU

Иван Пономарев

DSL тестовых данных

Декларативность

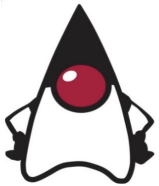
DSL тестовых данных



```
{
 "username": null,
 "password": null,
 "museum": {
   "name":  "Лувр Абу-Даби",
   "description": "Супер музей в ОАЭ",
   "city": "Абу-Даби",
   "country": "Объединённые Арабские Эмираты",
   "paintings": [
     {
       "name": "Автопортрет",
       "description": "Автопортрет Винсента Ван Гога",
       "artist": "Винсент ван Гог"
     }
   ]
 }
}
```

DSL тестовых данных

```
@User(
    museum = @Museum(
        name = "Лувр Абу-Даби",
        description = "Супер музей в ОАЭ",
        city = "Абу-Даби",
        country = "Объединённые Арабские Эмираты",
        paintings = @Paintings({
            @Painting(
                name ="",
                description ="",
                artist =""
            )
        })
    )
)
@Test void museumNameShouldEditedByAuthorizedUser()
```



Выводы

- Вы **точно можете** сделать подобное послезавтра на своем проекте
- Берите rococo, если “надо с нуля”. Развивайте идеи
- Пишите *элегантные* и простые тесты, вам скажет спасибо будущее поколение AQA.

Послесловие: а можно что-то еще?

CI/CD

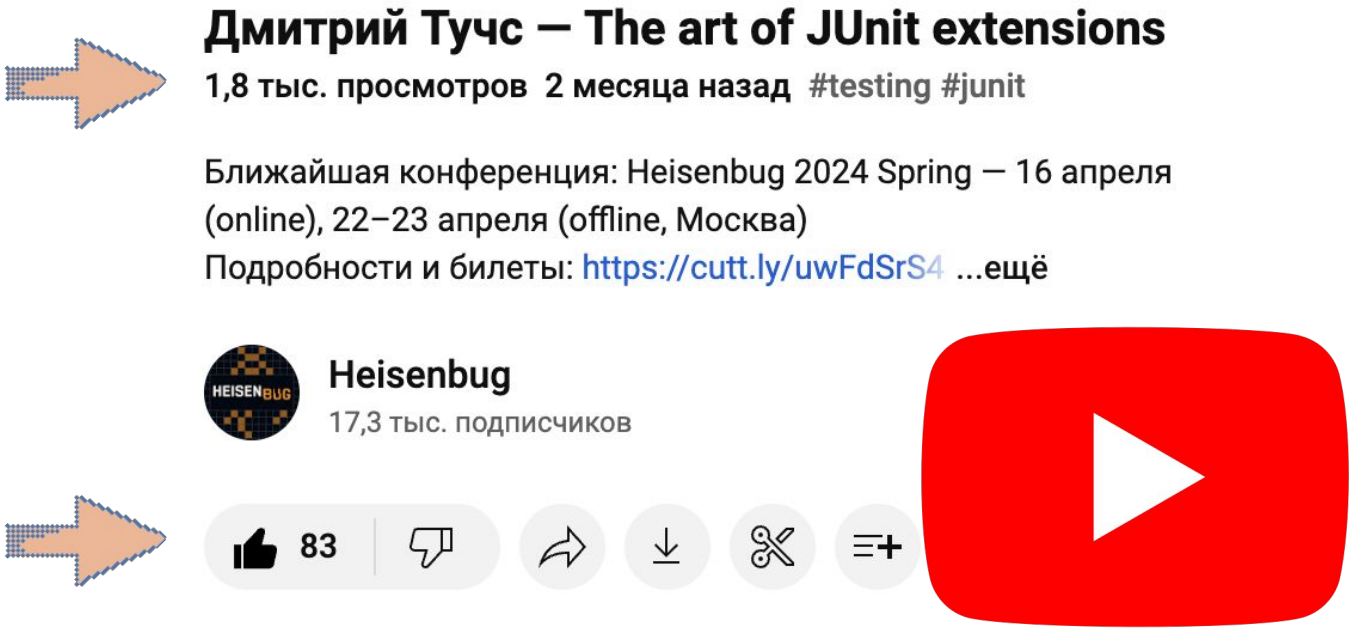
Docker / compose

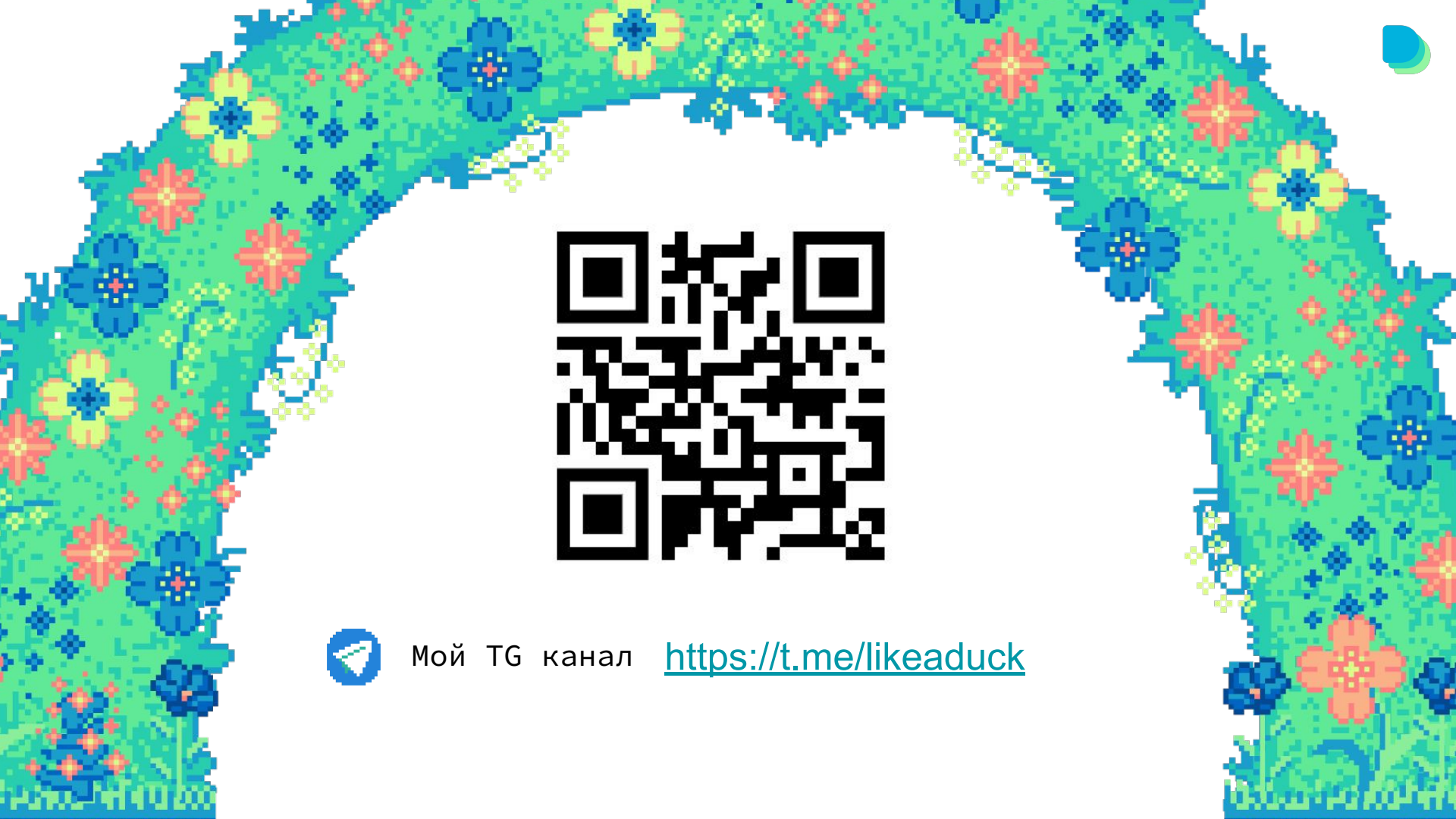
Работа с БД …

Ну и DSL с рефакторингом

*я бы быстро прощелкал этот квадратик если б уложился в час

А будет? Зависит от Вас :)





Мой TG канал https://t.me/likeaduck