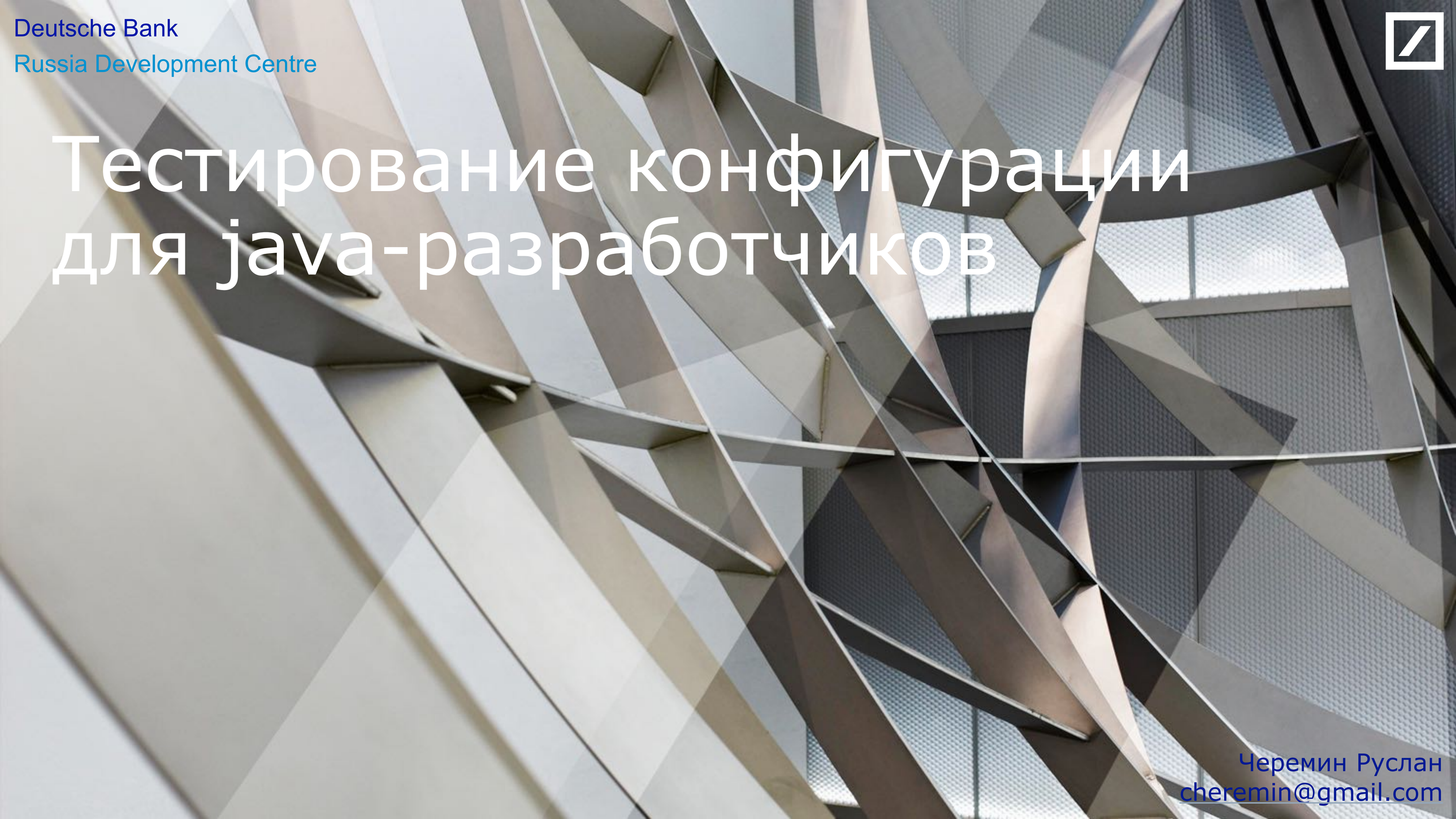
Deutsche Bank
Russia Development Centre



Тестирование конфигурации для java-разработчиков

Черемин Руслан
cheremin@gmail.com

Deutsche Bank
Russia Development Centre

Тестирование конфигурации для java-разработчиков

Черемин Руслан
cheremin@gmail.com

Safe harbour

Данный материал не является предложением или предоставлением какой-либо услуги. Данный материал предназначен исключительно для информационных и иллюстративных целей и не предназначен для распространения в рекламных целях. Любой анализ третьих сторон не предполагает какого-либо одобрения или рекомендации. Мнения, выраженные в данном материале, являются актуальными на текущий момент, появляются только в этом материале и могут быть изменены без предварительного уведомления. Эта информация предоставляется с пониманием того, что в отношении материала, предоставленного здесь, вы будете принимать самостоятельное решение в отношении любых действий в связи с настоящим материалом, и это решение является основанным на вашем собственном суждении, и что вы способны понять и оценить последствия этих действий. ООО "Дойче Банк Техцентр" не несет никакой ответственности за любые убытки любого рода, относящихся к этому материалу.

Простыми словами

- Все персонажи выдуманы
- Пользуйтесь с осторожностью
- Похороны за свой счет

Что такое “тестирование конфигурации”?

Есть важный код

```
src/main/java/com/db/app/
                    GainMoney.java
                    …
```

Для важного кода есть тесты

```
src/main/java/com/db/app/
                    GainMoney.java
src/test/java/com/db/app/
                    GainMoneyTest.java
                    …
```

Есть конфигурация

```
src/main/java/com/db/app/
                    GainMoney.java
src/test/java/com/db/app/
                    GainMoneyTest.java
src/main/resources/
                    app-lab.properties
                    app-uat.properties
                    app-prod.properties
```

Почему ее важно тестировать?

- ошибки конфигурации вредят бизнесу так же, как и ошибки кода

- конфигурация не проверяется компилятором/IDE
  - а часто недостаточно проверяется и при использовании

- UAT не гарантирует корректности PROD-конфигурации
  - и даже в PROD многие ошибки проявляются не сразу

Тесты для конфигурации:

This page
was intentionally left blank

HEISENBUG
// 2017 Moscow

Андрей Сатарин
Яндекс

Как проверить систему, не запуская её

Общий план

- Что можно успеть до обеда в понедельник
  - простые полезные примеры, “так можно делать”
- Понедельник, два года спустя:
  - где и как можно сделать лучше
- Поддержка для рефакторинга конфигурации
  - как добиться плотного покрытия
  - программная модель конфигурации

“Так делать — можно”

JMX ports conflict issue

```
env-uat.properties:
    configchanger.jmx.port    = 18501
    …
    spotserver.jmx.port       = 18503
    …
    republisher.jmx.port      = 18504
    …
    ratefan.jmx.port          = 18505
    …
    newservice.jmx.port       = 18505
```

JMX ports conflict issue

```
@Test
public void jmxPortsAreUnique() {
  File configsFolder = new File(CONFIGS_FOLDER);

  for( File configFile : configsFolder.listFiles(…) ) {
    Properties config = load( configFile );
    List<String> ports = filterValues(
      config,
      name -> name.contains( "jmx.port" )
    );

    assertThat( ports, itemsAreUnique() );
  }
}
```

JMX ports conflict issue

```
@Test
public void jmxPortsAreUnique() {
  File configsFolder = new File(CONFIGS_FOLDER);

  for( File configFile : configsFolder.listFiles(…) ) {
    Properties c
    List<String>
      config,
      name -> na
    );

    assertThat( ports, itemsAreUnique() );
  }
}
```

~~WTF~~А что, так можно было?

Ports are all valid

```
@Test
public void portsAreValid() {
  File configsFolder = new File(CONFIGS_FOLDER);
  for( File configFile : configsFolder.listFiles(…) ) {
    Properties config = load( configFile );
    List<String> ports = filterValues(
      config,
      name -> name.endsWith( ".port" )
    );

    assertThat( ports,
                eachItem(intValue(is(validNetworkPort()))))
    );
  }
}
```

Что проверяем здесь?

```
@Test
public void testNameObscured() {
  File configsFolder = new File(CONFIGS_FOLDER);
  for( File configFile : configsFolder.listFiles(…) ) {
    if( configFile.getName().contains( “prod" ) ) {
      Properties config = load( configFile );
      List<String> passwords = filterValues(
        config,
        name -> name.contains( ".passw" )
      );
      assertThat( passwords, everyItem(is(placeholder())));
    }
  }
}
```

Что проверяем здесь?

```
@Test
public void testNameObscured() {
  File configsFolder = new File(CONFIGS_FOLDER);
  for( File configFile : configsFolder.listFiles(…) ) {
    jdbc.password = ${very.secret.password}
        onfig,
        name -> name.contains( ".passw" )
      );
      assertThat( passwords, everyItem(is(placeholder())));
    }
  }
}
```

— А что считать конфигурацией?

— А что считать конфигурацией?

— Да все, что хочется

Пример: EOF для SQL-plus

```
14365-insert-important-data.sql:
   …
   INSERT INTO … VALUES …
   …
   /
   <<EOF>> must be on new line!
```

Пример: EOF для SQL-plus

```
@RunWith( Theories.class )
public class SQLPlusScriptsTest {
  @Theory
  public void scriptHasCorrectEnding( File sqlFile ) {
    String sql = Files.toString( sqlFile, US_ASCII );

    assertThat( sql, endsWith( "\n" ) );
    assertThat( sql.trim(), endsWith( "/" ) );
  }

  @DataPoints
  public static File[] sqlScripts() {
    return SQL_PATCHES_FOLDER.listFiles( ( dir, name ) ->
      name.endsWith( ".sql" )
    );
  }
}
```

Пример: crontabs

lab-sg.crontab:

```
0 12 * * 0 …/startAllServices.sh

0 10 * * 6 …/stopAllServices.sh
```

Пример: crontabs

lab-sg.crontab:

```
0 12 * * 0 …/startAllServices.sh

0 10 * * 6 …/stopAllServices.sh
<<EOF>>
```

Пример: crontabs

```
lab-sg.cron
```

```
@Theory
schedulesAreValid(File crontab)
```

```
0 12 * * 0 …/startAllServices.sh

0 10 * * 6 …/stopAllServices.sh
<<EOF>>
```

```
@Theory
EOF_isOnNewLine(File crontab)
```

Пример: shell-скрипты

env-**uat-us**.sh:

```
…
JAVA_HOME=…
LD_LIBRARY_PATH=…
AGGRESSIVE=1

…
```

Пример: shell-скрипты

```
env-u
   …
   JAVA_HOME=…
   LD_LIBRARY_PATH=…
   AGGRESSIVE=1
   …
```

```
@Theory
javaHomeIsDefined(File envFile)
```

```
@Theory
NDALib_IsInLD_LIBRARY_PATH(File envFile)
```

Итого

- Тесты конфигурации поначалу смущают
- Потом пишутся **легко и весело**
  - Много easy wins/low hanging fruits
- Уменьшают **затраты** на обнаружение и исправление ошибок конфигурации
- Дарят вторую молодость

Per-file tests

```
.../resources
        app-lab-uk.properties
        app-uat-us.properties
        app-prod-uk.properties
```

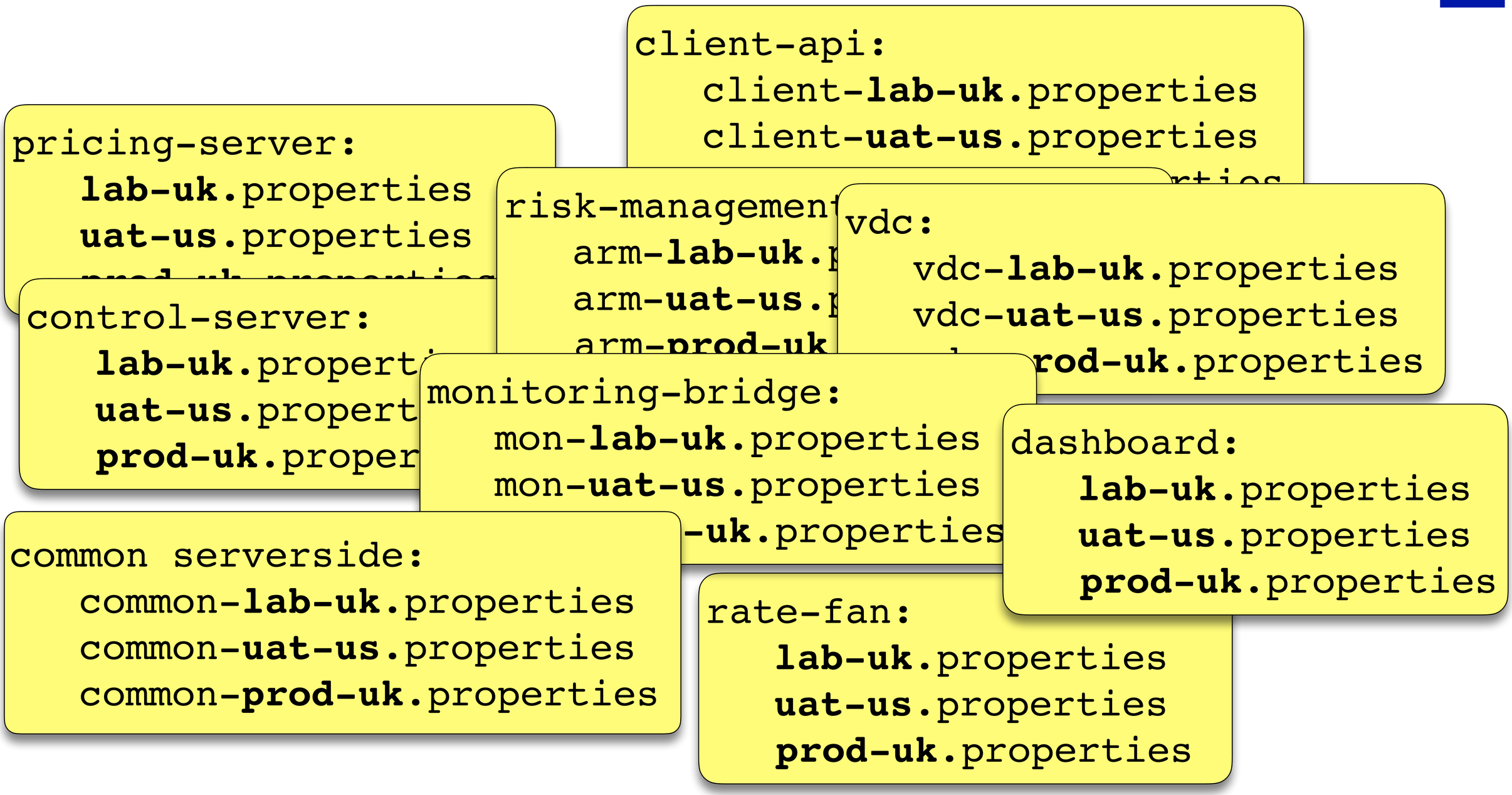
```
@Theory
forEachFileSomethingIsTrue(File envFile)
```

Сценарии посложнее

- UI-приложение соединяется с сервером **своего** environment-a

- Все сервисы одного environment-a соединяются с **одним и тем же** management-сервером

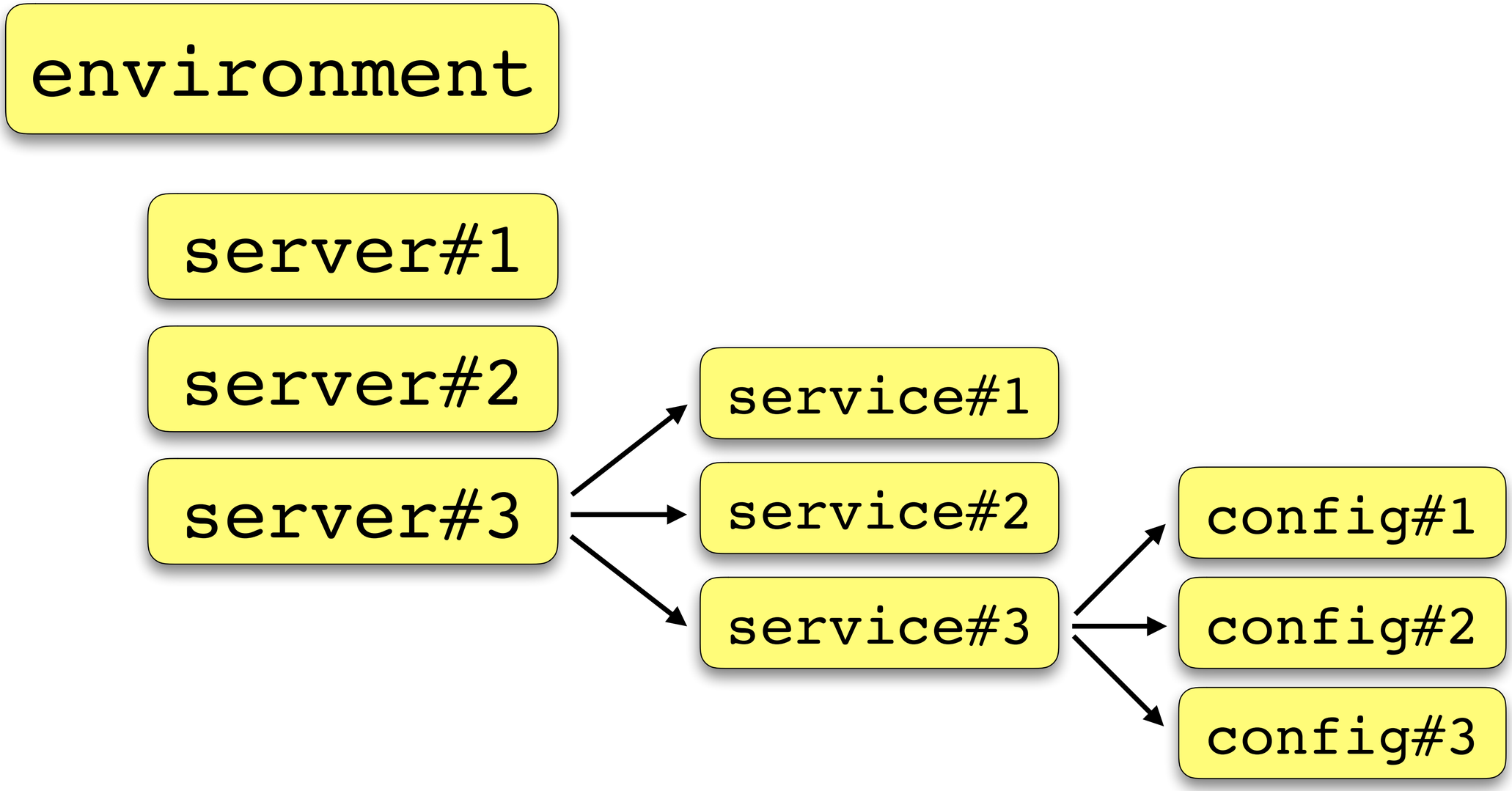
- Все сервисы одного environment-a используют **одну и ту же** базу данных


client-api:
client-lab-uk.properties
client-uat-us.properties
pricing-server:
lab-uk.properties
uat-us.properties
vdc:
vdc-lab-uk.properties
vdc-uat-us.properties
control-server:
monitoring-bridge:
mon-lab-uk.properties
mon-uat-us.properties
dashboard:
lab-uk.properties
uat-us.properties
prod-uk.properties
common serverside:
common-lab-uk.properties
common-uat-us.properties
common-prod-uk.properties
rate-fan:
lab-uk.properties
uat-us.properties
prod-uk.properties

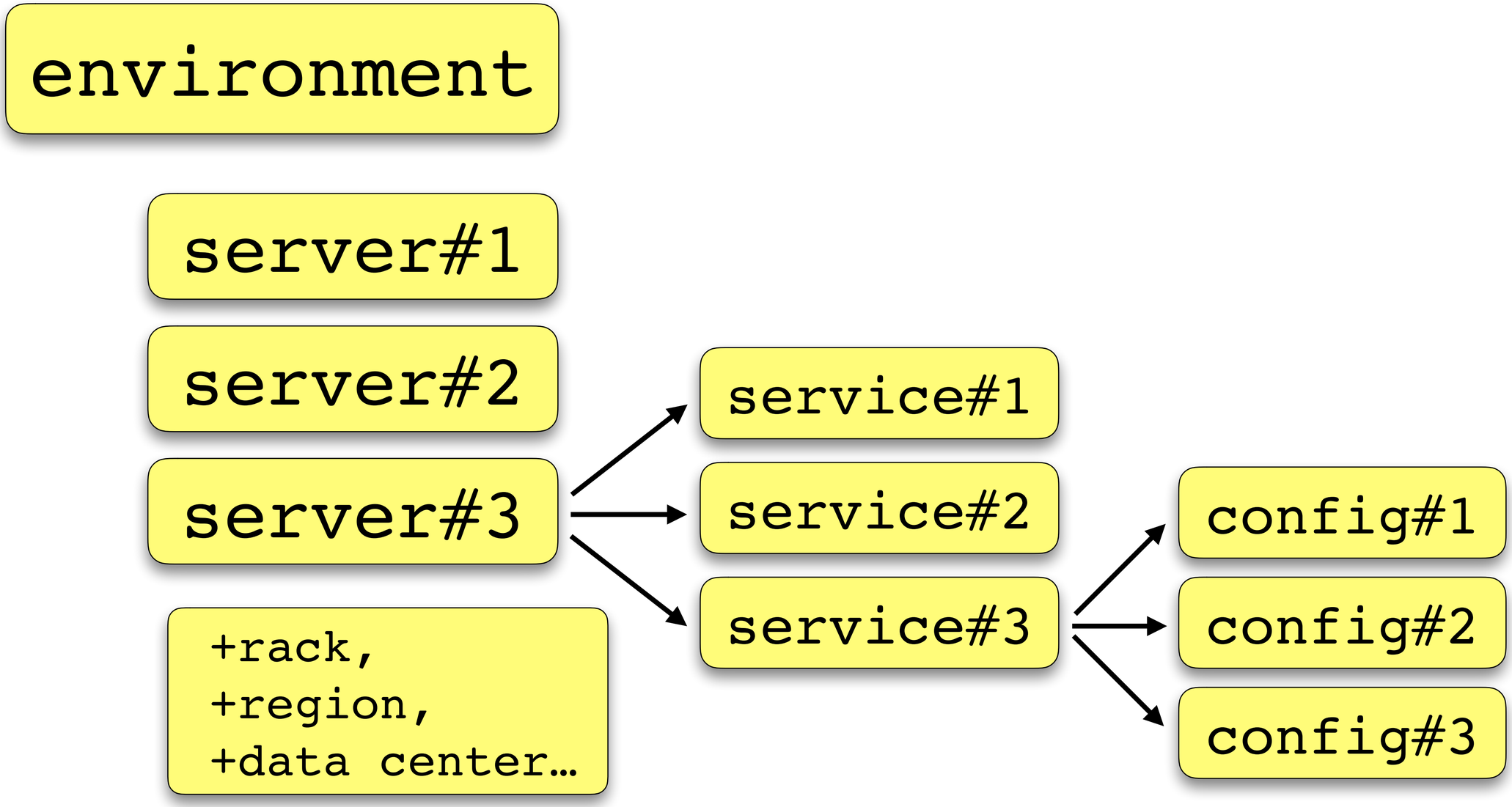
Как было бы удобнее?

```
@Theory
public void eachEnvironmentIsXXX( Environment environment ) {
  for( Server server : environment.servers() ) {
    for( Service service : server.services() ) {
      Properties config = buildConfigFor(
          environment,
          server,
          service
      );
      //… check {something} about config
    }
  }
}
```

Deployment layout



Deployment layout



Где взять deployment layout?

1. Сначала — захардкодить

2. Потом (может быть когда-нибудь) заинтегрироваться с существующей системой IM

Deployment layout

```
public enum Environment {
  PROD( PROD_UK_PRIMARY, PROD_UK_BACKUP,
        PROD_US_PRIMARY, PROD_US_BACKUP,
        PROD_SG_PRIMARY, PROD_SG_BACKUP )
  …

  public Server[] servers() {…}
}

public enum Server {
  PROD_UK_PRIMARY(“rflx-ldn-1"), PROD_UK_BACKUP("rflx-ldn-2"),
  PROD_US_PRIMARY(“rflx-nyc-1"), PROD_US_BACKUP("rflx-nyc-2"),
  PROD_SG_PRIMARY(“rflx-sng-1"), PROD_SG_BACKUP("rflx-sng-2"),

  public Service[] services() {…}
}
```

Deployment layout

```
public enum Environment {
   PROD( PROD_UK_PRIMARY, PROD_UK_BACKUP,
         PROD_US
         PROD_SG
   …

   public Server
}

public enum Se
   PROD_UK_PRIMA
   PROD_US_PRIMA
   PROD_SG_PRIMA

   public Service[] services() {…}
}
```

```
startServices-prod-uk-bcp.sh:

   …
   $START control-server
   $START pricing-server
   $START rate-fan

   …
```

Приятные бонусы на сдачу

```
public enum Environment {
  PROD( PROD_UK_PRIMARY, PROD_UK_BACKUP,
        PROD_US_PRIMARY, PROD_US_BACKUP,
        PROD_SG_PRIMARY, PROD_SG_BACKUP )

  public Service[] services() {…}
}
```

```
@Theory
eachEnvironmentHasCoreServices(Environment)

@Theory
criticalServicesHaveBackupInstances(Environment)
```

Поддерживаем актуальность

```
public class HardCodedLayoutConsistencyTest {
  @Theory
  eachHardCodedEnvironmentHasConfigFiles(Environment env){
    …
  }

  @Theory
  eachConfigFileHasHardCodedEnvironment(File configFile){
    …
  }
}
```

Итого: deployment layout

- Упрощает написание сложных тестов

- Делает их нагляднее и читабельнее

- В ходе его создания выясняются многие сакральные знания о deployment-е

- Хорошо дополняет документацию (особенно если ее нет)

Проблема

```
assertThat(
        filterProperties(
            properties,
            name -> name.contains(".port")
        ),
        eachItem(intValue(is(validNetworkPort())))
);
```

```
-> Error: 123456 is not valid network port
```

Нет контекста ошибки

```
assertThat(
        filterProperties(
            properties,
            name -> name.contains(".port")
        ),
        eachItem(in                                  Port()))))
);
```

file, propertyName?..
Чинить — где?

```
-> Error: 123456 is not valid network port
```

“Декларативные” тесты

```
SELECT
    environment, server, component, configLocation,
    propertyName,
    propertyValue
FROM
    configuration(environment, server, component)
WHERE
    propertyName like “%.port%”
    and
    propertyValue is not validNetworkPort()
```

“Декларативные” тесты

```
SELECT
    environment, server, component, configLocation,
    propertyName,
    propertyValue
FROM
    configuration(environment, server, component)
WHERE
    propertyName like “%.port%”
    and
    propertyValue is not validNetworkPort()
```

“Декларативные” тесты

```
SELECT
    environment, server, component, configLocation,
    propertyName,
    propertyValue
FROM
    configuration(environment, server, component)
WHERE
    propertyName like “%.port%”
    and
    propertyValue is not validNetworkPort()
```

“Декларативные” тесты

```
SELECT
    environment, server, component, configLocation,
    propertyName,
    propertyValue
FROM
    configuration(environment, server, component)
WHERE
    propertyName like “%.port%”
    and
    propertyValue is not validNetworkPort()
```

Use Streams, Luke

```
ValueWithContext[] incorrectPorts =
  flattenedProperties( environment )
        .filter( propertyNameContains( ".port" ) )
        .filter(
           !isInteger( propertyValue )
           || !isValidNetworkPort( propertyValue )
        )
        .toArray();

assertThat( incorrectPorts, emptyArray() );
```

Use Streams, Luke

```
ValueWithContext[] incorrectPorts =
  flattenedProperties( environment )
      .filte( propertyNameContains( ".port" ) )
      .filte
         !is                                  e )
         ||                                   propertyValue )
      )
      .toArr

assertThat(                                    Array() );
```

```
ValueWithContext {
    environment,
    server,
    service,
    configPath,
    propertyName,
    propertyValue
}
```

Use Streams, Luke

```
ValueWithContext[] incorrectPorts =
  flattenedProperties( environment )
        .filter( propertyNameContains( ".port" ) )
        .filter(
            !isInteger( propertyValue )
            || !isValidNetworkPort( propertyValue )
        )
        .toArray();

assertThat( incorrectPorts, emptyArray() );
```

Use Streams, Luke

```
ValueWithContext[] incorrectPorts =
  flattenedProperties( environment )
      .filter( propertyNameContains( ".port" ) )
      .filter(
          !isInteger( propertyValue )
          || !isValidNetworkPort( propertyValue ) )

assertThat( incorrectPorts, emptyArray() );
```

```
Expected: Empty array
     got: [<PROD/PROD_UK/vdc:vdc-prod-uk.properties:vdc.jmx.port = 123456>]
```

Тестирование как поддержка для рефакторинга

С чем имеем дело

```
xxx.address=${${hostname}.bus.address}
…
xxx.name=${Seq-${hostname}}.${${hostname}.shortname}
…
xxx.mcast=${Seq.udp.${seq.${hostname}.order}.recv.mcast}

# и так до самого низа…
```

С чем имеем дело

```
apps/sequencer.properties
   include logging.properties
      include logging-${OS}.properties
   include network.properties
   include admin.properties
   include ports.properties
      include env/${env}.properties
         include hosts/${host-1}.properties
         include hosts/${host-2}.properties
         include hosts/${host-3}.properties
```

Конфигурация 1 сервиса

10 `include`-ов глубиной **3**

~450 параметров всего

Нужны этому сервису только **10-15%**

* Автор — гений, но его с нами больше нет

Хочется

- Конфигурацию отрефакторить и упростить

Хочется

- Конфигурацию отрефакторить и упростить

- После рефакторинга каждый сервис имеет все **необходимые** параметры

Хочется

- Конфигурацию отрефакторить и упростить

- После рефакторинга каждый сервис имеет все **необходимые** параметры

- …и не имеет **лишних** параметров

Хочется

- Конфигурацию отрефакторить и упростить

- После рефакторинга каждый сервис имеет все **необходимые** параметры

- …и не имеет **лишних** параметров

- Сервисы успешно соединяются в кластер

Хочется

- Конфигурацию отрефакторить и упростить

- После рефакторинга каждый сервис имеет все **необходимые** параметры

- …и не имеет **лишних** параметров

- Сервисы успешно соединяются в кластер

- (…но неизвестно, соединяются ли они сейчас!)

Программа-минимум

- Проверить корректность каждого параметра каждого сервиса в изоляции

- Проверить ключевые взаимосвязи между параметрами

Как это протестировать?

```
public class SimpleComponent {
  …
  public void configure( final Configuration conf ) {
    int port = conf.getInt( "Port", -1 );
    if( port < 0 ) throw new ConfigurationException();

    String ip = conf.getString( "Address", null );
    if( ip == null ) throw new ConfigurationException();
    …
  }
  …
}
```

Как это протестировать?

```
public class S…ponent {
  …
  public void configure( final Configuration conf ) {
    int port = conf.getInt( "Port", -1 );
    if( port < 0 ) throw new ConfigurationException();

    S…  conf.getString( "Address", null );
    if( … ull ) throw new ConfigurationException();
    …
  }

  …
}
```

type

name

default value

validation

Самодостаточные свойства

```
IProperty<Integer> PORT_PROPERTY
    = intProperty( "Port" )
        .withDefaultValue( -1 )
        .matchedWith( validNetworkPort() );

IProperty<String> ADDRESS_PROPERTY
    = stringProperty( "Address" )
        .withDefaultValue( null )
        .matchedWith( validIPAddress() );
```

Самодостаточные свойства

```
interface IProperty<T> {
  /* (name, defaultValue, matcher…) */

  /** lookup (or use default),
    * convert type,
    * validate value against matcher
    */
  FetchedValue<T> fetch( final Configuration config )
}

class FetchedValue<T> {
  public final String propertyName;
  public final T propertyValue;
  …
}
```

Configuration model

```
class SimpleComponentModel implements IConfigurationModel{
  IProperty<Integer> PORT = intProperty("Port")
                .withDefaultValue( -1 )
                .matchedWith( validNetworkPort() );

  IProperty<String> ADDRESS = stringProperty("Address")
                .withDefaultValue( null )
                .matchedWith( validIPAddress() );

  @Override
  List<FetchedValue<?>> fetch(Configuration config){
    return asList( PORT.fetch(config), ADDRESS.fetch(config ));
  }
}
```

Модели комбинируемы

```
class ComplexComponentModel implements IConfigurationModel{
  final IConfigurationModel subComponent1 = …;
  final IConfigurationModel subComponent2 = …;


  @Override
  List<FetchedValue<?>> fetch(Configuration config){
    return union(
          subComponent1.fetch(config),
          subComponent2.fetch(config)
    );
  }
}
```

Чего это стоило

- 12 сервисов

- 70 конфигурируемых классов
  => 70 `ConfigurationModels` (~60 тривиальны)

- **1-2 человеко-недели**

Что получается

```
@Theory
public void propertiesValidInIsolation( Environment environment ){
  for( Server server : environment.servers() ) {
    for( Service service : server.services() ) {
      Configuration config = fetchConfigFor(
          environment,
          server,
          service

      );
      IConfigurationModel model = service.configurationModel();
      model.fetch( config );
    }
  }
}
```

Взаимозависимости сервисов

Большинство зависимостей это сетевые связи:

“Сервис А слушает именно тот адрес,
на который отправляет пакеты сервис Б”

Network endpoints

```
IProperty<IEndpoint> TCP_REQUEST = outcomingTCP(
    // (+matchers, +default values)
    “TCP.Request.Address”,
    “TCP.Request.Port”
);

class OutcomingTCPEndpoint implements IEndpoint {
    //(localInterface, localAddress, multicastGroup, port)

    @Override
    boolean matches( IEndpoint other);
}
```

Проверка связности кластера

```
ValueWithContext[] allEndpoints =
    flattenedConfigurationValues(environment)
        .filter( valueIsEndpoint() )
        .toArray();

ValueWithContext[] unpairedEndpoints =
    Arrays.stream( allEndpoints )
        .filter( e -> !hasMatchedEndpoint(e, allEndpoints) )
        .toArray();

assertThat( unpairedEndpoints, emptyArray() );
```

И швейцарский нож на сдачу…

ConfigurationModel позволяет:

- Конвертировать в другой формат

- Выполнять запросы к конфигурации
  (“все udp-порты, используемые сервисами на данном сервере”)

- Экспортировать сетевые связи между сервисами в виде диаграммы

Выводы

- Тестировать конфигурацию **важно**
- Поле непаханое, а порог входа низкий
  - От самых простых тестов уже много пользы
  - Тестировать сложные взаимосвязи тоже возможно
- На простых тестах возможности не заканчиваются
  - Можно решать и сложные задачи
  - Получая на сдачу полезные инструменты

Что дальше?

Дальше пока не придумал, импровизируй

Для жалоб и предложений

Руслан Черемин

cheremin@gmail.com, ruslan.cheremin@db.com

blog: dev.cheremin.info