FLYING START INTO CONTRACT TESTING

Piotr Kubowicz @pkubowicz
$ whois piotr-kubowicz

- Java/JVM since Dec 2007
- Software developer @ Ocado Technology Kraków
AGENDA

1. Purpose of Contract Tests
2. Writing using Pact
3. Use cases
4. Comparing Pact with Spring Cloud Contract
Planner app

How many forklifts to move supplies arriving tomorrow?

Supplies system

- tomorrow: 300 apples, 45 kg
- tomorrow: 40 bottles of water, 60 kg
- +2 days: 600 carrots, 36 kg
TESTING WITH REAL DEPENDENCIES

('integration testing')
TYPICAL PROBLEM

```json
{
    "count": 15,
    "status": "CANCELED"
}
```
CONTRACT TESTING

enforced in an automatic way
TERMINOLOGY

Consumer \[\rightarrow\text{calls}\] \rightarrow \text{Provider}
LET'S SEE IT WORKING
SPEED

Consumer: 220 test cases per minute
Provider: 2400 test cases per minute
WORKFLOW

1. Consumer test
2. Play with API
3. Share contract
4. Adopt Provider
WORKFLOW

No waiting until Provider features are ready
You can start without Provider itself
SHARING PACT CONTRACTS

Attach to an e-mail then commit to Git
Publish to your artifact repository (Maven)
Dedicated Pact Broker server
LANGUAGES SUPPORTED

- JVM
- JS
- Ruby
- .NET
- Go
- Swift
- PHP
- any (for provider)
PACT ON THE PROVIDER SIDE

- Pact Provider
- Verifier
- Docker
- Your Provider
- any language
PACT ON THE PROVIDER SIDE

Planner

Shop

Supplies
WHAT TO TEST

no assertions

```kotlin
@Test @PactTestFor(pactMethod = "twoSupplies")
fun getsSupplies() {
    client.getFor(LocalDate.of(2018, 12, 23))
}
```
WHAT TO TEST

only check client

```kotlin
val received = client.getFor(LocalDate.of(2018, 12, 23))
assertThat(received).hasSize(2)
assertThat(received[1])
    .hasFieldOrPropertyWithValue("count", 4)
    .hasFieldOrPropertyWithValue("status", ACTIVE)
    .hasFieldOrPropertyWithValue("totalWeight", 20)
```
WHAT TO TEST

check client and logic

```kotlin
val averageWeight =
    SuppliesAnalyser().countAverageItemWeight(
        client.getFor(LocalDate.of(2018, 12, 23))
    );
assertThat(averageWeight).isEqualTo(5.0);
```
WHAT TO TEST

isolated client contract test

```java
var supplies = client.getFor(LocalDate.of(2018, 12, 23))
assertThat(supplies.get(0))
    .hasFieldOrPropertyWithValue("status", ACTIVE)
```

isolated logic test

```java
var supplies = listOf(
    Supply(2, 12, ACTIVE), Supply(3, 13, ACTIVE))
assertThat(analyser.countAverageItemWeight(supplies))
    .isEqualTo(5.0)
```
WHAT TO TEST

logic implementation

```kotlin
val sumWeights = supplies.map { it.totalWeight }.sum()
```

requirement: canceled supplies not counted

```kotlin
var supplies = listOf(
    Supply(2, 12, ACTIVE),
    Supply(3, 13, ACTIVE))
assertThat(analyser.countAverageItemWeight(supplies))
    .isEqualTo(5.0)
```
CONTROLLING PROVIDER BEHAVIOUR

test 1

.aponReceiving("request for a canceled supply")

test 2

.aponReceiving("request for two supplies")
CONTROLLING PROVIDER BEHAVIOUR

/supplies?day=2018-12-23 → one canceled supply
/supplies?day=2018-12-01 → one canceled and one active supply
CONTROLLING PROVIDER BEHAVIOUR

/current-user-details
→ {"admin": false}

/current-user-details
→ {"admin": true}
test 1

```
given("one canceled supply")
uponReceiving("request for a canceled supply")
```

test 2

```
given("one canceled and one active supply")
uponReceiving("request for two supplies")
```
SPECIFYING BODY

```
.body(newJsonBody { o ->
    o.stringValue("status", "CANCELED")
})
```

Missing required creator property 'count' (index 0)
at [Source: (String)"[{"status":"CANCELED"}]"; line: 1, column: 18
SPECIFYING BODY

Consumer: return 50

Provider: verify 50 is returned
SPECIFYING BODY

Consumer: return 50

Provider: verify any number is returned
SPECIFYING BODY

```
o.stringMatcher("checksum", "\w{8}", "a421ef07")
```
CONTRACT TESTS USE CASES

- backend-frontend in one team
- teams in different locations
- project handover
- message queues
### Pacts

<table>
<thead>
<tr>
<th>Consumer</th>
<th>Provider</th>
<th>Latest pact published</th>
<th>Webhook status</th>
<th>Last verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orders</td>
<td>Delivery</td>
<td>about 1 hour ago</td>
<td>Create</td>
<td>18 minutes ago</td>
</tr>
<tr>
<td>Planner</td>
<td>Employees</td>
<td>about 1 hour ago</td>
<td>Create</td>
<td></td>
</tr>
<tr>
<td>Planner</td>
<td>Supplies</td>
<td>11 minutes ago</td>
<td>Create</td>
<td>less than a minute ago</td>
</tr>
<tr>
<td>Shop</td>
<td>Orders</td>
<td>11 minutes ago</td>
<td>Create</td>
<td></td>
</tr>
<tr>
<td>Shop</td>
<td>Supplies</td>
<td>about 1 hour ago</td>
<td>Create</td>
<td>less than a minute ago</td>
</tr>
</tbody>
</table>

5 pacts
VISUALISATION
A pact between Planner and Supplies

Requests from Planner to Supplies

- Request for a canceled supply
- Request with malformed date

Interactions

Upon receiving request for a canceled supply from Planner, with

```json
{
  "method": "GET",
  "path": "/supplies",
  "query": "day=2018-12-23"
}
```

Supplies will respond with:

```json
{
  "status": 200,
  "headers": {
```
ALTERNATIVE TOOLS

Spring Cloud Contract
CONTRACT IN SPRING CLOUD CONTRACT

```java
org.springframework.cloud.contract.spec.Contract.make {
    request {
        method 'GET'
        url('/supplies') {
            queryParameters {
                parameter day: '2018-12-23'
            }
        }
    }
    response {
        status OK()
        body([status: "CANCELED"])
        headers { contentType('application/json') }
    }
}
```
Generated test

```java
public class ManySuppliesTest extends ContractVerifierBase {
    @Test
    public void validate_returnsManySupplies() throws Exception {
        Response response = given().spec(request)
            .queryParams("day","2018-12-23").get("/supplies");
        // ...
        assertThatJson(parsedJson).array()
            .contains("['status']").isEqualTo("CANCELED");
    }
}
```

Base class you need to write

```java
abstract class ContractVerifierBase {
    @Before
    fun startServer() {
        RestAssured.port = 9053
        MyServer().start(9053)
    }
}
```
• Test 1: request for two supplies
• Test 2: request for a canceled supply
response {
    body([{
        status: "CANCELED",
        count: 4, totalWeight: 20, status: "ACTIVE"},
    ])
}

Accepted response

[
    {"status": "CANCELED"},
    {"count": 4, "totalWeight": 20, "status": "ACTIVE"},
    {"count": 151, "totalWeight": 980, "status": "ACTIVE"}
]
response {
    body([[
    [status: "CANCELED"],
    [count: 4, totalWeight: 20, status: "ACTIVE"],
    ]])
}
Spring Cloud Contract can work with Pact contracts
NO TIME FOR THIS — I'M TOO BUSY

- writing tests for code sending HTTP requests
- asking other team what data they send
- thinking about edge cases
- testing my DB queries return what I want
CONTRACT TESTS

- test correct form of Consumer requests
- document requests/responses
- share test data and edge cases
- verify full stack of Provider
Q & A

piotr.kubowicz@gmail.com

Twitter @pkubowicz

github.com/pkubowicz/contract-testing-samples

pact.io

cloud.spring.io/spring-cloud-contract/
ATTRIBUTION

Photo by dylan nolte on Unsplash

Icons used under CC BY 3.0 license are made by

- iconfinder.com/user/atifarshad
- iconfinder.com/im04
- iconfinder.com/CreativeCorp

Explosion icon by iconfinder.com/korawan_m

Question mark and cake icon by iconfinder.com/Field5